# Diplomarbeit

Thema:

# Spracherkennung im Chinesischen

Jürgen Reichert

Betreuer: Prof. Dr. A. Waibel
Dipl. Inform. Tanja Schultz
Fakultät: Informatik
Zeitraum: 1.7.98 – 31.12.98
Institut: für Logik, Komplexität und
Deduktionssysteme
der Universität Karlsruhe

Ich erkläre hiermit, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Karlsruhe, den 24.11.1998

# Vorwort

Die sprecherunabhängige Spracherkennung kontinuierlicher, spontaner Sprache mit großem Vokabular ist gegenwärtig ein wichtiges Forschungsgebiet. Die „Spracherkennung im Chinesischen“, mit der sich die vorliegende Arbeit beschäftigt, ist von besonderer Bedeutung, da im Chinesischen die Texteingabe mittels Tastatur einen unvergleichbar höheren Aufwand als in westlichen Ländern darstellt. Die Spracherkennung, als das natürlichste Kommuniktionsmedium, ist besonders geeignet diese Aufgabe zu erfüllen.
Ziel dieser Arbeit ist, unter Berücksichtigung der besonderen Eigenschaften der chinesischen Sprache, geeignete Ansätze zur Spracherkennung zu untersuchen und einen leistungsfähigen Spracherkenner aufzubauen. Ein weiterer Schwerpunkt dieser Arbeit ist die Portierung des Janussystems auf das Windows-Bertriebsystem, sowie die Entwicklung einer grafischen Benutzerschnittstelle für das Janussystem und weiterer Hilfsmittel zur Arbeitserleichterung.

Die vorliegende Arbeit ist meinen Eltern und meiner lieben Frau Ping gewidmet, ohne deren fachliche Unterstützung die Arbeit in dieser Art und Weise nicht möglich gewesen wäre.

An dieser Stelle möchte ich meiner Betreuerin Dipl. Inform. Tanja Schultz für die gute Zusammenarbeit und für viele wertvolle Ratschläge danken. Mein Dank gilt auch Herrn Prof. A.Waibel der die Arbeit in dieser Form ermöglichte.

# Inhaltsverzeichnis

**Vorwort** .......... **3**

**Inhaltsverzeichnis** .......... **4**

**Abbildungsverzeichnis** .......... **7**

**1 Einleitung** .......... **10**

1.1 Aufgabenstellung .......... 12

1.2 Motivation .......... 13

1.3 Gliederung der Arbeit .......... 14

**2 Grundlagen automatischer Spracherkennung** .......... **15**

2.1 Sprachproduktion und Sprachperzeption beim Menschen .......... 15

2.2 Aufbau eines Spracherkenners .......... 18

2.2.1 Vorverarbeitung und Digitalisierung .......... 18

2.2.2 Merkmalsextraktion .......... 19

2.2.3 Merkmalstransformation .......... 21

2.2.4 Klassifikation .......... 21

**3 Die chinesische Sprache** .......... **31**

3.1 Entwicklung der Zeichen .......... 31

3.2 Morphologie der chinesischen Sprache .......... 36

3.3 Das gesprochene Chinesisch .......... 37

3.4 Weitere Besonderheiten der chinesischen Sprache .......... 40

3.4.1 Die Schreibrichtung .......... 40

3.4.2 Die Phonetik und Tonalität .......... 41

3.4.3 Das chinesisches Zahlensystem .......... 44

3.4.4 Die Grammatik der chinesischen Sprache .......... 44

3.4.5 Sprichwörter (成语) .......... 44

3.4.6 Sonstiges .......... 46

3.5 Chinesisch und Computer .......... 46

3.5.1 Eingabemethoden ....46
3.5.2 Zeichendarstellung und Fonts ....47
3.5.3 Codierung ....47
3.5.4 Vorteile der chinesischen Schrift ....48
3.5.5 Chinesische Systeme ....49

**4 Das Spracherkennungswerkzeug Janus ....50**

4.1 Einführung in Janus ....50
4.2 Portierung nach Windows ....54
4.3 Benutzerinterface ....54
4.3.1 Projektverwaltung ....55
4.3.2 Skripte ausführen und editieren ....57
4.3.3 Erzeugen der Konfigurationsdateien für Janus ....58
4.3.4 Skriptausführung ....66
4.3.5 Debugger und hierarchischer Editor für Tcl/Tk ....66
4.3.6 Audio- und Transkriptionsbearbeitung ....69
4.3.7 Audiokonvertierung ....72
4.3.8 Die Audioschnittstelle ....74
4.3.9 Die Perl-Umgebung ....76
4.3.10 Hilfesystem ....77

**5 Chinesische Spracherkennung ....78**

5.1 Überblick über Ansätze chinesischer Spracherkennung ....78
5.2 Die Datensammlung ....79
5.3 Aufbereitung der Daten ....80
5.4 Erzeugung der Konfigurationsdateien ....83
5.4.1 Aufteilung der Daten ....84
5.4.2 Definition des Vokabulars ....84
5.4.3 Definition des PhonemSet ....87
5.4.4 Erzeugung des Aussprachewörterbuchs ....91
5.4.5 Konstruktion des Languagemodels ....92
5.4.6 Vorverarbeitung und HMM ....95
5.4.7 Initialisierung des chinesischen Erkenners ....97

5.5 Training des kontextunabhängigen Erkenners ....97
5.5.1 Labels schreiben ....97
5.5.2 LDA berechnen ....98
5.5.3 Samples extrahieren ....98
5.5.4 Kmeans berechnen ....99
5.5.5 Trainingslauf ....99
5.5.6 Test ....99
5.6 Aufbau eines kontextabhängigen Erkenners ....102
5.7 Aufbau eines Silbenerkenners ....104
5.8 Erzeugung von chinesischen Zeichen aus der Pinyinumschrift ....105
5.9 Gesamtperformance des chinesischen Erkenners ....109

**6 Zusammenfassung und Ausblick ....110**

**Anhang A: Verschiedenes ....111**

**Anhang B: Objekte in Janus ....113**

**Literaturverzeichnis ....125**

# Abbildungsverzeichnis

Abbildung 2.1: Schematische Darstellung der Sprachproduktion beim Menschen..................15
Abbildung 2.2: Quellenfiltermodell........................................................................................16
Abbildung 2.3: Phon-Skala.....................................................................................................17
Abbildung 2.4: Bark-Skala .....................................................................................................17
Abbildung 2.5: Grundmodell eines Spracherkenners .............................................................18
Abbildung 2.6: Analoges Signal der Äußerung “ta”................................................................18
Abbildung 2.7: Digitales Signal der Äußerung “a1”................................................................19
Abbildung 2.8: Leistungsspektrum der Äußerung “a1” ...........................................................19
Abbildung 2.9: Spektrogramm der Äußerung “ta1 yi4 hui4” ...................................................20
Abbildung 2.11: Architektur des MS-TDNN ..........................................................................24
Abbildung 2.12: Forward-Backward-Trellis............................................................................26
Abbildung 2.13: Forward-Backward-Reestimation..................................................................27
Abbildung 2.14: HMM - Topologien.......................................................................................28
Abbildung 2.15: Viterbi-Suche................................................................................................30
Abbildung 3.1: Erste Schriftfunde auf Tonscherben in Banpo ...............................................31
Abbildung 3.2: Entwicklung von der frühen Bilderschrift zu heutigen chinesischen Zeichen.32
Abbildung 3.3: Orakelknocheninschrift...................................................................................34
Abbildung 3.4: Bronzeinschrift der Zhou-Dynastie..................................................................34
Abbildung 3.5: Siegelinschrift .................................................................................................35
Abbildung 3.6: Offizielle Schrift wie sie heute noch in Taiwan verwendet wird.....................35
Abbildung 3.7: Vereinfachte Schrift wie sie heute in der VR China verwendet wird..............36
Abbildung 3.8: Die Verteilung der Dialekte in China (aus [13])............................................39
Abbildung 3.9: sechs Phoneme des zeichenbasierten Bopomofo-Systems ..............................39
Abbildung 3.10: Beispiel einer in Taiwan gebräuchlich Schreibrichtungen ............................41
Abbildung 3.11: Tabelle der Pinyin - Sprachlaute des Mandarin Dialekts...............................42
Abbildung 4.1: Janus-Shell in einem X-Window unter Windows ...........................................51
Abbildung 4.2: Janusobjekte und deren Abhängigkeiten ........................................................52
Abbildung 4.3: JanusUI – grafische Benutzeroberfläche für Janus..........................................55
Abbildung 4.4: JanusUI Dialogbox zum Öffnen eines Systems..............................................56

Abbildung 4.5: JanusUI Dialogbox zum Einstellen von Parametern ......................................56

Abbildung 4.6: Janus Console ........................................................................................57

Abbildung 4.7: JanusUI Symbolleiste.............................................................................57

Abbildung 4.8: Dialog zur Parametereingabe..................................................................58

Abbildung 4.9: Description-Menü ..................................................................................58

Abbildung 4.10: Erzeugung der Tags-Datei ...................................................................59

Abbildung 4.11: Erzeugung des PhonesSets...................................................................59

Abbildung 4.12: Erzeugung der DBase ..........................................................................60

Abbildung 4.13: Festlegen der HMM-Topologien ..........................................................61

Abbildung 4.14: Erzeugen der Stream-Dateien ..............................................................62

Abbildung 4.15: Beispiel DistribTree..............................................................................63

Abbildung 4.16: Definition der Vorverarbeitung und des Zugriffspfades.......................63

Abbildung 4.17: Aufteilung der Sprecher in Trainings-, Test-, Evaluationsmenge ................64

Abbildung 4.18: Phonemmapping ..................................................................................65

Abbildung 4.19: TDebug ................................................................................................66

Abbildung 4.20: Outline-Debugger .................................................................................67

Abbildung 4.21: Play and Record - Tool .........................................................................70

Abbildung 4.22: Fontauswahl..........................................................................................71

Abbildung 4.23: Audioformat-Konverter ........................................................................72

Abbildung 4.24: Audioformat wählen .............................................................................73

Abbildung 4.25: Wave/Feature-Betrachter......................................................................74

Abbildung 4.26: Perlumgebung.......................................................................................76

Abbildung 4.27: Startseite des Hilfesystems ..................................................................77

Abbildung 5.1: Aufnahmeorte in der VR China ............................................................80

Abbildung 5.2: Zuordnung von Sprechern zu Herkunftsorten.........................................80

Abbildung 5.3: Funktionsweise des Pinyinumsetzers......................................................81

Abbildung 5.4: Grafische Oberfläche des Pinyinumsetzers.............................................83

Abbildung 5.5: Erzeugung des Trainingsvokabulars .......................................................85

Abbildung 5.6: Erzeugung der Liste häufig vorkommender Worte.................................86

Abbildung 5.7: Erzeugung der Menge aller Aussprachesilben........................................86

Abbildung 5.8: Zusammensetzung des Trainings- und Testvokabulars ...........................86

Abbildung 5.9: Erzeugung der Menge aller Aussprachesilben........................................87

Abbildung 5.10: Tabelle aller Phoneme mit IPA-Entsprechung......................................89

Abbildung 5.11: IPA-Konsonantenschema....................................................................90
Abbildung 5.12: IPA-Vokalschema..........................................................................90
Abbildung 5.13: Erzeugung des Ausspracheswörterbuchs........................................................91
Abbildung 5.14: Überdeckung des Korpus durch die n-häufigsten Worte ..............................93
Abbildung 5.15: Feature-Betrachter aus der Janus Toolssammlung........................................96
Abbildung 5.16: Codebuch/Distribution-Betrachter aus der Janus Toolssammlung...............96
Abbildung 5.17: Zuordnungspfad-Analyse aus der Janus Toolssammlung.............................98
Abbildung 5.18: Veranschaulichung der verschiedenen Suchpässe ......................................100
Abbildung 5.19: Performanceentwicklung ..............................................................104
Abbildung 5.20: Rücktransformation der Pinyinumschrift in chinesische Zeichen ..............105
Abbildung 5.21: Erzeugung der Umsetzungsregeln ..................................................106
Abbildung 5.22: Performancedaten des phonembasierten chinesischen Spracherkenners.....109
Abbildung 5.23: Performancedaten des silbenbasierten chinesischen Spracherkenners ........109
Abbildung A.1: Windows Explorer ............................................................111

# 1 Einleitung

Die automatische Spracherkennung gewinnt aus vielen Gründen immer mehr an Bedeutung. Zum einen ist erstmals genügend Rechenleistung zu einem akzeptablen Preis verfügbar und zum anderen sind die existierenden Diktiersysteme heutzutage schon so weit fortgeschritten, daß sie den reinen Spielzeugstatus hinter sich gelassen haben, auch wenn sie die eine oder andere Kinderkrankheit noch nicht vollständig überwunden haben. (Ein Teil der vorliegende Arbeit wurde dem Diktiersystem Via Voice 4.3 Gold von IBM diktiert [61], wobei die Erkennungrate nach längerem Training über 95% lag, aber noch Verbesserungen in der Benutzeroberfläche nötig wären.) Außer in reinen Diktierlösungen werden Spracherkennungssysteme ebenfalls in vielfältigen anderen Anwendungsgebieten eingesetzt. Dabei stehen dann andere Anforderungen wie z.B. eine spezielle Domäne, die Integration in andere Umgebungen, die Robustheit, die Kosten der Plattform oder die einfache Bedienung im Vordergrund, so daß sich insgesamt ein sehr breites Spektrum von Anwendungsmöglichkeiten ergibt:

- **Diktierlösungen:** Diktat von Geschäftsbriefen, Reports oder Analysen. Hier ist es von Vorteil, daß die Hände für andere Aufgaben frei sind, was z.B. für Ärzte von grundlegender Bedeutung ist. Für behinderte Menschen ergeben sich dadurch ganz neue Möglichkeiten zur Kommunikation. Für Gehörlose können automatisch Filmuntertitelungen generiert werden. Analphabeten werden in die Lage versetzt, ohne fremde Hilfe Briefe zu schreiben.
- **Informationsysteme:** Informationen können mittels der natürlichsten Kommunikationsform, der Sprache, erlangt werden. Dadurch kann eine etwaige Hemmschwelle vor der Computerbedienung abgebaut werden.
- **Bedienung von elektrischen Geräten:** Mittels Sprache können Geräte wie HiFi-Anlagen, Videorecorder usw. intuitiv bedient werden. Die Bedienung von Funktelefonen oder Navigationssytemen im Fahrzeug kann ohne Blickkontakt und somit risikoärmer erfolgen.
- **Call-center:** Der Einsatz von automatischer Spracherkennung in telefonischen Auskunftssystemen erfährt momentan den größten Auftrieb, da sich der Kundenservice, bei gleichzeitiger Kosteneinsparung, erheblich verbessern läßt. Beispielsweise werden solche Systeme schon bei Hotlineanbietern, Zug-/Flugverbindungssystemen, bei Bank- und Börsengeschäften, Terminabsprachen, Buchungs- und Reservierungssystemen, Preisauskunftssystemen und Telefonvermittlungsanlagen zur automatischen Generierung einer Durchwahl eingesetzt.
- **Übersetzungssysteme:** In Verbindung mit automatischer Übersetzung und Sprachsynthese ist es möglich in gewissen Situationen auf einen Dolmetscher verzichten zu können, so daß sich Personen miteinander verständigen, ohne eine gemeinsame Sprache beherrschen zu müssen. Das Sprach-zu-Sprach-Übersetzungssystem Janus [24] der Universität Karlsruhe und der Carnegie Mellon University ermöglicht dies mittels mehrerer Sprachen auf kleineren Domänen wie z.B. Terminabsprache. Dabei erfolgt die Übersetzung in beiden Richtungen in jeweils 3 Schritten: Spracherkennung → Textübersetzung → Sprachsynthese.
- **Lernprogramme:** Da sich Multimediaanwendungen mit Sprachausgabe immer mehr durchsetzen, ist es nur konsequent, Eingaben auch mittels Sprache vornehmen zu können. Besonders vorteilhaft ist dies bei Fremdsprachenlernprogrammen, da diese somit auch die Aussprache des Lernenden überwachen können.
- **Sprach- und Sprecheridentifikation:** Manchmal ist notwendig, neben dem was gesprochen wurde, auch ermitteln zu können, wer oder in welcher Sprache etwas gesprochen wurde. Ein Spezialeinsatzgebiet ist die Sprecherverifikation, mit der festgestellt werden kann, ob der Sprecher auch derjenige ist, für den er sich ausgibt.

- **Überwachungssysteme:** Für polizeiliche, geheimdienstliche oder militärische Abhörmaßnahmen werden schon seit Jahren Systeme, die auf Schlüsselwörter reagieren, eingesetzt. So können wichtige von unwichtigen Informationen getrennt werden. Diese Trennung macht auch Sinn, um gewisse Angebote aus Radio und Fernsehen gezielt zu selektieren.

Genauso vielfältig wie die Anwendungsgebiete der automatischen Spracherkennung sind die Forschungsgebiete, die sich mit Spracherkennung und Sprachverarbeitung befassen. Die Signalaufnahme und Akustik ist ein Teilgebiet der Physik. Die elektrische Signalverarbeitung, wie Verstärkung, Filterung und Digitalisierung gehört in den Bereich der Elektrotechnik. Die Algorithmen der Mustererkennung, Informationstheorie und Methoden der künstlichen Intelligenz sind ein Metier der Informatiker, zu dem die Mathematiker ihren Anteil in Form von mathematischen Verfahren der Statistik und Funktionentheorie beisteuern. Die Linguisten decken wiederum die wichtigen Bereiche Phonetik, Morphologie, Grammatik, Semantik und Pragmatik ab. Die Psychologen und Philosophen fügen weiteres, die Hintergründe der Sprache betreffendes Wissen, hinzu. Und schließlich liefern die Biologen und Neurologen wichtige Kenntnisse über die Sprachproduktion, Sprachrezeption und deren Weiterverarbeitung beim Menschen, dem Vorbild der automatischen Spracherkennung. Das Ziel ist also eine ähnliche uneingeschränkte und freie Kommunikation zwischen Mensch und Maschine, wie zwischen zwei Menschen. Wobei allerdings innerhalb der Spracherkennung das Sprachverstehen nur soweit von Interesse ist, wie es dem Erkennungsvorgang behilflich ist.

Obwohl schon mehrere Jahrzehnte auf dem Gebiet der automatischen Spracherkennung geforscht wurde, existiert noch keine alle Aspekte befriedigende Lösung. Wenn auch für uns die Sprache etwas ganz natürliches ist, ist die maschinelle Spracherkennung – im Vergleich zur Eingabe per Tastatur – mit großen Schwierigkeiten behaftet, deren Ursachen die folgenden Punkte umfaßt [50]:

- Kontinuität: Bei kontinuierlich gesprochener Sprache können Wortgrenzen nicht mehr detektiert werden, ohne die ganze Äußerung betrachten zu müssen. Auch die Bestimmung von Silben- und Phonemgrenzen ist keine triviale Aufgabe.
- Spontanität: In nicht gelesenen Äußerungen treten oft Füllwörter ohne Bedeutung wie äh, oh usw. auf. Versprecher, Phrasenwiederholungen und grammatische Fehler erschweren eine grammatische Analyse sehr. Wortneuschöpfungen und Modeausdrücke werden situationsbezogen verwendet, so daß sich die gesprochene von der Schriftsprache stark unterscheidet.
- Variabilität: Aufgrund verschiedener Akzente und Sprechweisen ist es möglich ein und dieselbe Äußerung auf eine unterschiedliche Art und Weise zu sprechen. Verschiedene Sprechgeschwindigkeiten, Betonungen, Koartikulationen oder das Verschlucken ganzer Silben beeinflussen weiterhin das Sprachsignal. Verschiedene Sprecher haben aufgrund ihres Alters, ihres Geschlechts, ihres Gesundheitszustandes, ihrer Körperfülle, sowie spezieller Eigenheiten eine individuelle Aussprache, die es uns ermöglicht eine Person zu identifizieren, es dem Computer aber sehr erschwert, die herausragenden Merkmale verschiedener Aussprachen eines Wortes zu extrahieren. Selbst eine Person hat in verschiedenen Situationen und aufgrund ihres Gemütszustandes unterschiedliche Aussprachen.
- Störquellen: Störgeräusche des Sprechers, wie Husten, Atmen, Lippengeräusche oder Hintergrundgeräusche der Umgebung, müssen von dem Nutzsignal unterschieden werden. Desweiteren muß die Aufnahmecharakteristik von Mikrofonen und die Akustik des Aufnahmeortes berücksichtigt werden.
- Komplexität: Mit der Vokabulargröße nimmt auch der benötigte Rechenaufwand und Speicherbedarf explosionsartig zu, da man bei einer Äußerung mit n Worten auf einem Vokabular aus v Worten im worst case $v^n$ zu betrachtende Hypothesen erhält. Neben der

Größe des Suchraums nimmt auch die Verwechselbarkeit von Worten mit der Mächtigkeit des Vokabulars zu.

- Homophone: Das sind Wörter mit derselben Aussprache, aber verschiedenen Bedeutungen (z.B. weg – Weg, das – daß, Rad – Rat). Um Homophone richtig zuordnen zu können, reicht die reine akustische Information nicht aus. Weitere Informationsquellen wie Syntax, Semantik, Pragmatik und Prosodie müssen hinzugezogen werden.

Diese Aufzählung kann einen Eindruck davon vermitteln, wie komplex doch das Gebiet der automatischen Spracherkennung ist. Es ist sicher nicht möglich alle aufgezählten Punkte auf einmal befriedigend erfüllen zu wollen. Aus diesem Grund haben sich in der Forschung verschiedene Richtungen herausgebildet, die unterschiedliche Schwerpunkte einzeln betrachten und andere Aspekte entsprechend einschränken. Als mögliche Einschränkungen haben sich dabei ergeben: Statt kontinuierlicher Sprache nur pausenbegrenzte Worte zuzulassen (Einzelworterkenner), die Vokabulargröße auf einem bestimmten Anwendungsfall zu beschränken, nur eine einfache Grammatik zuzulassen, das System nur auf einen Sprecher zu trainieren oder auf einen Sprecher zu adaptieren, Störgeräusche zu minimieren oder eine Kooperation des Sprechers zu erwarten.

## 1.1 Aufgabenstellung

Die vorliegende Diplomarbeit „Spracherkennung im Chinesischen“ untersucht Konzepte, um einen Spracherkenner, unter Berücksichtigung der Besonderheiten der chinesischen Sprache aufzubauen. Dabei stützen sich die Untersuchungen auf den Spracherkennungsteil des Sprach-zu-Sprach-Übersetzungssystems Janus [53]. Dieses wurde im laufe der Arbeit von Unix auf das Betriebssystem Windows portiert und um ein Anzahl von Hilfsprogrammen erweitert.

Die chinesische Sprache ist in vielerlei Hinsicht von den uns geläufigen Sprachen verschieden. Dies kommt im Volksmund durch Redewendungen wie „das kommt mir aber chinesisch vor“ oder „dies ist Fachchinesisch für mich“ zum Ausdruck. Tatsächlich enthält sowohl die Schriftsprache als auch gesprochene Sprache Eigenschaften, die eine besondere Betrachtungsweise erfordern. Diese Arbeit untersucht diese Eigenschaften der chinesischen Sprache, soweit sie die Spracherkennung betreffen näher, und versucht Möglichkeiten zu finden, um die haupt-sächlich für europäische Sprachfamilien entwickelten Sprachverarbeitungsverfahren auch auf die chinesische Sprache anwenden zu können. Die erste dieser Eigenschaften ist, daß es in der chinesischen Schrift keine Wortgrenzen gibt. Für eine wortbasierte Verarbeitung müssen Wortgrenzen somit künstlich eingeführt werden und zwar so, daß die entstehenden Teilabschnitte Sinneinheiten repräsentieren. Dazu müssen die kleinsten Sinneinheiten eines Satzes gefunden werden, wobei es aufgrund grammatikalischer und semantischer Besonderheiten nicht trivial ist diese Trennung vorzunehmen. In vielen Fällen wird auch pragmatisches Wissen für die Trennung benötigt, und oftmals ist eine von Hand durchgeführte Partitionierung mit einem gewissen subjektiven Charakter behaftet, da es nicht immer eindeutige Regeln gibt. Eine weitere besondere Eigenschaft der chinesischen Sprache ist jene, daß es im Chinesischen keine direkte Bindung der Aussprache an die Schrift gibt, es also erforderlich wird, ein Verfahren zu entwickeln, das die zeichenbasierte Schriftsprache in eine Lautschrift überführt und die Lautschrift auch wieder zurück in die Schriftsprache abbilden kann. Diese Abbildungen sind nicht trivial und können nur aus einer großen Menge von Trainingsdaten trainiert werden. Außergewöhnlich im Vergleich zu europäischen Sprachen ist auch die Tatsache, daß wichtige sprachliche Informationen der chinesische Sprache in der Tönhöhenmodulation der Aussprache enthalten sind, so daß auch diese sonst unwichtigere Information betrachtet werden muß. Diese und weitere Besonderheiten wie grammatikalische und sprachkulturelle werden in einem eigenen Kapitel behandelt.

Neben diesen sprachspezifischen Gegebenheiten werden auch Hilfsmittel zum effizienten Aufbau eines Spracherkenners untersucht. Diese basieren auf der nach Windows portierten Klassenbibliothek und Skriptesammlung des Janussystems und sind in eine Benutzerschnittstelle integriert. Durch diese Integration soll der Einarbeitungsaufwand eines Erstbenutzers reduziert werden und Bedienfehler, die oftmals erst nach Stunden oder Tage zum Vorschein kommen, wenn möglich unterbunden werden. Dies geschieht dadurch, daß der Benutzer alle möglichen Optionen, die zu einem bestimmten Zeitpunkt relevant sind, in einem Dialogfenster angeboten bekommt, und er diese anpassen kann ohne fehlerträchtig Skriptfiles ändern zu müssen. Es geht dabei keine Flexibilität verloren, da die Skripte weiterhin transparent editiert werden können. Desweiteren ist ein Debugger in die Bedienumgebung integriert, so daß Fehler in Skripten aufgespürt werden können, die sonst nur mühevoll aufgedeckt werden könnten. Ein Projektmanagement und eine Projektablaufsteuerung nehmen dem Benutzer die Routinearbeit ab und helfen dabei, den Überblick über den Projektstand nicht zu verlieren. Viele dieser Tools entstanden, um den Umgang mit der Spracherkennung zu vereinfachen und die Entwicklung und den Einsatz eines Spracherkenners, basierend auf Janus, auch auf der weit verbreiteten Plattform Windows zu ermöglichen. Die Tools befinden sich in jeweils in unterschiedlichen Entwicklungsstadien, so daß mit Änderungen gerechnet werden muß.

## 1.2 Motivation

Die Relevanz, die heutzutage die automatische Spracherkennung im Allgemeinen hat, wurde oben bereits erwähnt. Aber in wie weit ist die Spracherkennung im Chinesischen überhaupt interessant?

Seit Mitte dieses Jahrhunderts bis zum Ende der achtziger Jahre hat China eine Politik der Isolation geführt. Bis auf wenige Ereignisse, wie die Befreiung/Machtergreifung durch die Kommunisten 1948 und die Kulturrevolution (Säuberung von politisch Andersdenkenden) 1966-76 ist China fast völlig im zeitpolitischen Geschehen untergegangen. Nur wenige beschäftigten sich mit chinesischer Politik, Wirtschaft und Kultur. Dies hatte zur Folge, daß es seit der Öffnung Chinas seit ca. 1979 einen großen Nachholbedarf an Informationen zu befriedigen galt. Denn mit zunehmender wirtschaftlicher und politischer Bedeutung Chinas wurde es immer wichtiger neue Beziehungen zu China aufzubauen.

Anlaß sich eingehend mit der chinesischen Sprache zu beschäftigen, gibt es somit genug:

- Wirtschaftliche, politische und kulturelle Bedeutung Chinas
- Chinesisch sprechen ca. 20% der Weltbevölkerung, somit ist Chinesisch die Sprache, die die meisten Sprecher hat. Auch in der westlichen Welt existieren Städte, in deren China-Towns mehrere zehntausend Menschen hauptsächlich chinesisch sprechen.
- Chinesisch ist diejenige lebendige Sprache, deren Entwicklungsgeschichte am weitesten zurückreicht. Somit belegt Chinesisch für die verschiedenen Sprachwissenschaften von besonderer Bedeutung.
- Viele literarische Werke liegen in keiner Übersetzung vor. Auch moderne Fachliteratur in Forschung und Wissenschaft, sowie Recherchedatenbanken, Statistiken und Gesetzesregelungen sind oftmals nur in der Landessprache erhältlich. Literatur zu den, dem westlichen Schach entfernt ähnelnden, Spielen Weiqi (Go) und Xiangqi (chin. Schach), die auch bei uns eine stark wachsende Anhängerschaft finden, ist bei uns nur schwer zu finden. Besonders Weiqi hat in der Forschung der künstlichen Intelligenz dem westlichen Schach den Rang abgelaufen.
- Kalligraphie, die Fähigkeit des ausdrucksvollen Schreibens mit dem Pinsel, ist ein fester Bestandteil der chinesischen Kunst. Für ein tieferes Verständnis der chinesischen Kunst ist somit ein gewisses Gefühl für chinesische Schrift erforderlich.

- Der Tourismus in China entwickelt sich rasant. Neben Wirtschaftsreisen werden auch immer mehr Studien- und Erhohlungsreisen unternommen. Laut einer Prognose aus „China Contract 10/98“ soll sich China sogar ab 2010 weltweit zum Tourismusland Nummer 1, vor Frankreich und den USA, entwickelt haben. Für Reiseveranstalter, Studienenreisende, Geschäftsleute und Urlauber sind sicher einige Kenntnisse der chinesischen Sprache von Vorteil.
- Da für Europäer das Lernen asiatischer, zeichenbasierender Sprachen wie Chinesisch, Japanisch und Koreanisch (CJK), verglichen mit anderen europäischen Sprachen, äußerst schwierig ist, ist eine computerunterstützte Sprachverarbeitung von besonderer Wichtigkeit.
- Die Konzepte der chinesischen Sprache unterscheiden sich von denen unserer Sprache sehr, so daß Chinesisch ein guter Kandidat ist, um möglichst viele dieser Konzepte kennenzulernen. Insbesondere kann die Flexibilität des Janussystems mittels dieser Konzepte validiert werden.

## 1.3 Gliederung der Arbeit

Diese Arbeit beginnt im 2. Kapitel mit einer Einführung in die Grundlagen automatischer Spracherkennung. Es werden die grundsätzliche Ansätze eines aktuellen Spracherkenners erläutert. Aufbauend darauf werden die einzelnen Phasen und Prozesse jeweils in einem Unterkapitel näher betrachtet. Kapitel 3 befaßt sich mit der chinesische Sprache. Es verschafft einen Überblick über die strukturelle Entwicklung der chinesischen Sprache und wichtige Besonderheiten im Hinblick auf die Computerverarbeitung. Kapitel 4 führt in das Spracherkennungs-werkzeug Janus ein. Es werden die grundlegenden Module und Konzepte beschrieben. Darauffolgend werden einige Bemerkungen zur Portierung folgen, bevor die Benutzerschnittstelle und einige Hilfsprogramme dargestellt werden. Im 5. Kapitel wird der Aufbau eines chinesi- schen Spracherkenners basierend auf den zwei Ansätzen der Phonemerkennung und Silben-erkennung betrachtet und Experimente zur Erkennungsleistung der Spracherkenner präsentiert. Die Arbeit wird in Kapitel 6 zusammengefaßt und es werden mögliche weitere Entwicklungen diskutiert.

# 2 Grundlagen automatischer Spracherkennung

Die Forschungen auf dem Gebiet der automatischen Spracherkennung reichen fast fünfzig Jahre zurück. Erste Arbeiten befaßten sich mit der Entwicklung von Klangspektrographen zur visuellen Darstellung von Schallsignalen oder untersuchten die menschliche Spracheproduktion, um die Grundlagen für erste Phonem- oder Einzelworterkenner zu schaffen [50]. Nach ersten Anfangserfolgen stellten sich weitere Fortschritte nur langsam ein. Erst in den siebziger Jahren, als sowohl technische wie auch theoretische Voraussetzungen erfüllt waren, erfolgte eine kontinuierliche Verbesserung der maschinellen Spracherkennung von Jahr zu Jahr.

## 2.1 Sprachproduktion und Sprachperzeption beim Menschen

Da die Spracherkennung als die Umkehrung der Sprachproduktion angesehen werden kann, können durch eine nähere Untersuchung dieser wichtige Erkenntnisse gewonnen werden.

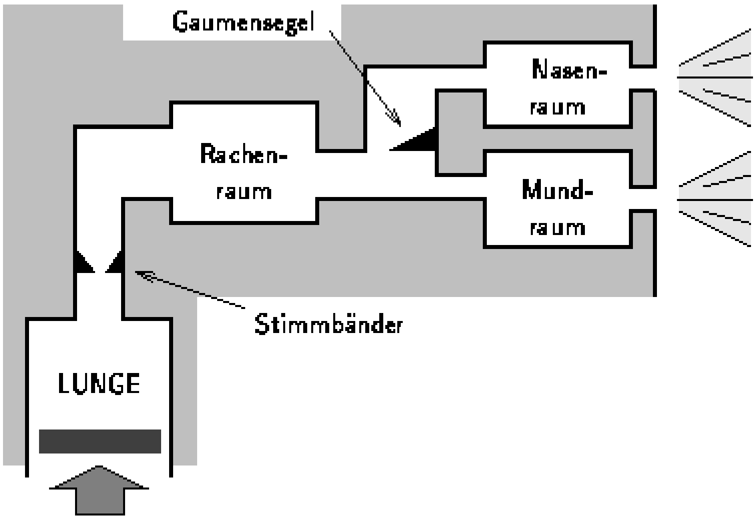


Abbildung 2.1: Schematische Darstellung der Sprachproduktion beim Menschen

Wie in obiger Abbildung dargestellt, sind mehrere Komponenten an der Erzeugung des Sprachsignals beteiligt. Dadurch daß Muskeln den Lungenraum zusammendrücken, entsteht ein Luftstrom, der die Stimmbänder (Glottis) zum Schwingen bringt. Für diese Schwingungen ist der Bernoulli-Effekt verantwortlich, der an der Oberfläche einer zum Luftstrom parallelen Fläche einen Unterdruck aufbaut, der wiederum eine Zugkraft auf die Fläche bewirkt. Bei den Stimmbändern bewirkt diese Kraft einen Verschluß derselben, so daß keine Luft mehr strömen kann und die Stimmbänder sich wieder öffnen. Die Frequenz dieses Rückkopplungseffekts kann durch die Muskelspannung an den Stimmbändern gesteuert werden. So können Luftschwingungen von einigen Hz bis zu mehreren kHz erzeugt werden. Das Resonanzverhalten der Hohlräume Rachen-, Nasen- und Mundraum beeinflußt die erzeugten Schwingungen. Der Nasenraum kann durch das Gaumensegel bei Bedarf (zum Erzeugen von Nasalen) hinzugeschaltet werden, so daß Antiresonanzen erzielt werden. Eine besondere Bedeutung kommt der Zunge, die den Mundraum in zwei Resonanzräume unterteilen kann, und den Lippen/Zähnen zu, die die Abstrahlung der Schallwellen beeinflussen können. G. Fant entwikkelte, abstrahierend von der menschlichen Sprachproduktion, das Quellenfiltermodell [11], indem er Stimmbänderdämpfung, Vokaltrakt und Abstrahlung als eine Aneinanderreihung von Filtern betrachtete.

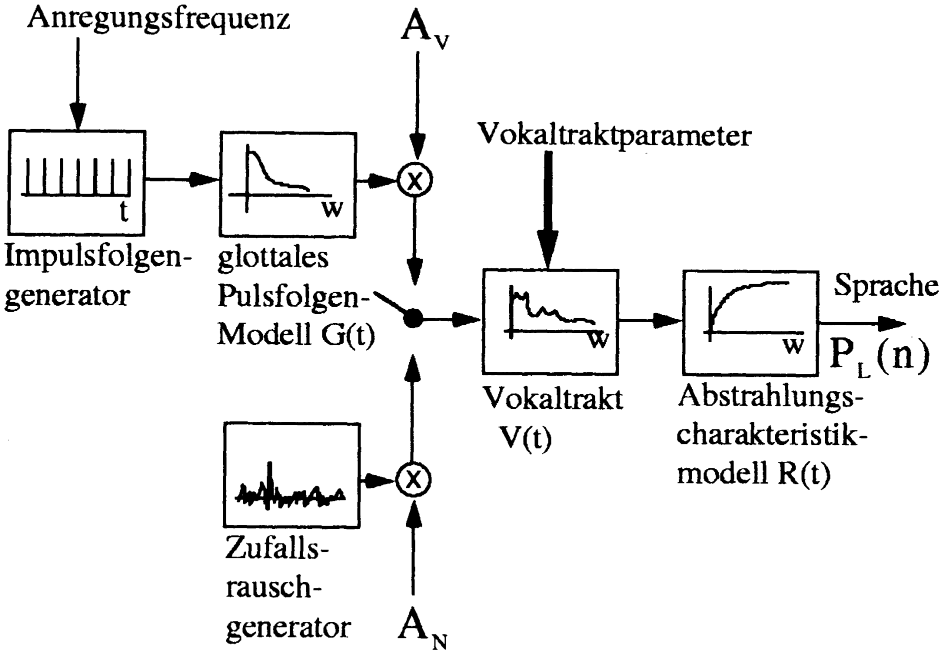


Abbildung 2.2: Quellenfiltermodell

Grundsätzlich werden zwei Arten der Spracherzeugung unterschieden. Bei stimmhaften Lauten wird die Luft durch die Stimmbänder in Schwingungen versetzt, während bei stimmlosen Lauten die Stimmbänder soweit geöffnet werden, daß der Bernoulli-Effekt wirkungslos bleibt, und nur ein Rauschen entsteht, welches durch den Vokaltrakt und die Abstrahlungscharakteristik weiter geformt wird.

Für die Spracherkennung ergeben sich hieraus interessante Ansätze. So kann für die Vokalerkennung der Erzeugungsprozeß im wesentlichen auf den Vokaltrakt reduziert und mittels eines parameterisierbaren Röhrenmodells nachgebildet werden, so daß eine einfache Übertragungsfunktion der Form

$$H(z) = \frac{\sigma}{\sum_{i=0}^{p} a_i z^{-i}}$$

den Vokaltrakt ausreichend annähert, solange keine Nasale gesprochen wurden. Wenn anhand des diskreten Sprachsignals die Parameter $\sigma$ und $a_i$ geschätzt wurden, kann auf das zugrunde liegende Röhrenmodell und somit auch den entsprechenden Vokal geschlossen werden. Die obige Formel hat eine sehr einfache Form, so daß durch die lineare Vorhersage (Linear Predictive Coding, LPC) ein effektives Schätzverfahren vorliegt [43].

Forschungen am menschlichen Gehör über die Wahrnehmung von Schallwellen haben ebenso wichtige Erkenntnisse erbracht. So ist der Hörbereich auf Frequenzen von ca. 16 –20000 Hz beschränkt, wobei die Schallempfindlichkeit im mittleren Frequenzbereich am höchsten ist. Um dieser Eigenart des menschlichen Gehörs gerecht zu werden, wird ein spezielles Maß das **phon** zur Beschreibung der Lautestärkeempfindung eingeführt. Die Phonzahl entspricht dabei der Leistungzahl in dB eines Tones bei 1 kHz. Die nachfolgende Abbildung gibt die Leistung gleichlaut empfundener Töne in [dB] an.

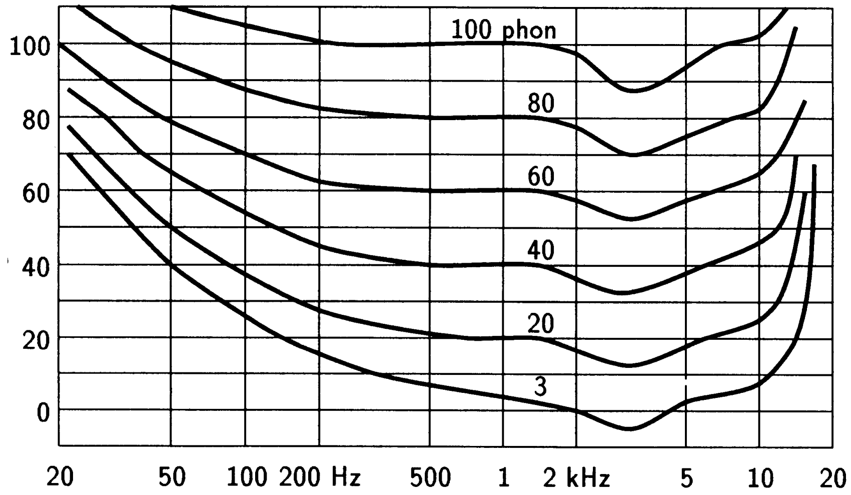


Abbildung 2.3: Phon-Skala

Neben dem frequenzabhängigen Lautstärkeeindruck des Gehörs, besteht auch eine Abhängigkeit der Frequenzauflösung von dem Frequenzbereich. So ist die Frequenzauflösung bei hohen Tönen wesentlich niedriger als bei tiefen. Diese Eigenschaft macht man sich in der Spracherkennung zunutze, indem man bei der Merkmalsextraktion Frequenzintervalle entsprechend wählt und somit eine Merkmalsreduktion durchführen kann. Eine Abbildung von Frequenzbereichen auf Frequenzbereiche mit gleichbleibender Frequenzunterscheidbarkeit geschieht durch die Bark-Skala (oder mel-Skala mit 1 bark = 100 mel).

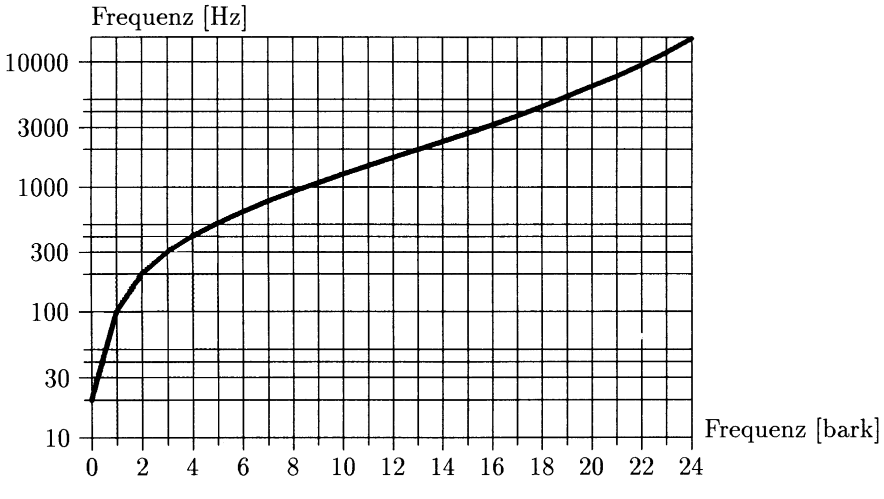


Abbildung 2.4: Bark-Skala

Ein weiterer Effekt, der beim menschlichen Gehör festgestellt werden kann, ist, daß das Ohr frequenz- und lautstärkeabhängige Einschränkungen bei der Auflösung von nahe beieinanderliegenden akustischen Ereignissen hat. So können starke Signale schwächere Signale in unmittelbarer zeitlicher Nähe überdecken. Diesen Effekt macht man sich vornehmlich bei Audiokompressionsverfahren wie MP3 zu nutze.

## 2.2 Aufbau eines Spracherkenners

Die Spracherkennung, wie sie heute gesehen wird, ist ein mehrstufiger Prozeß, der Schallwellen in eine Satzhypothese überführt. Dieser Prozeß ist in der folgenden Abbildung schematisch dargestellt und wird in den folgenden Unterkapiteln näher beschrieben.

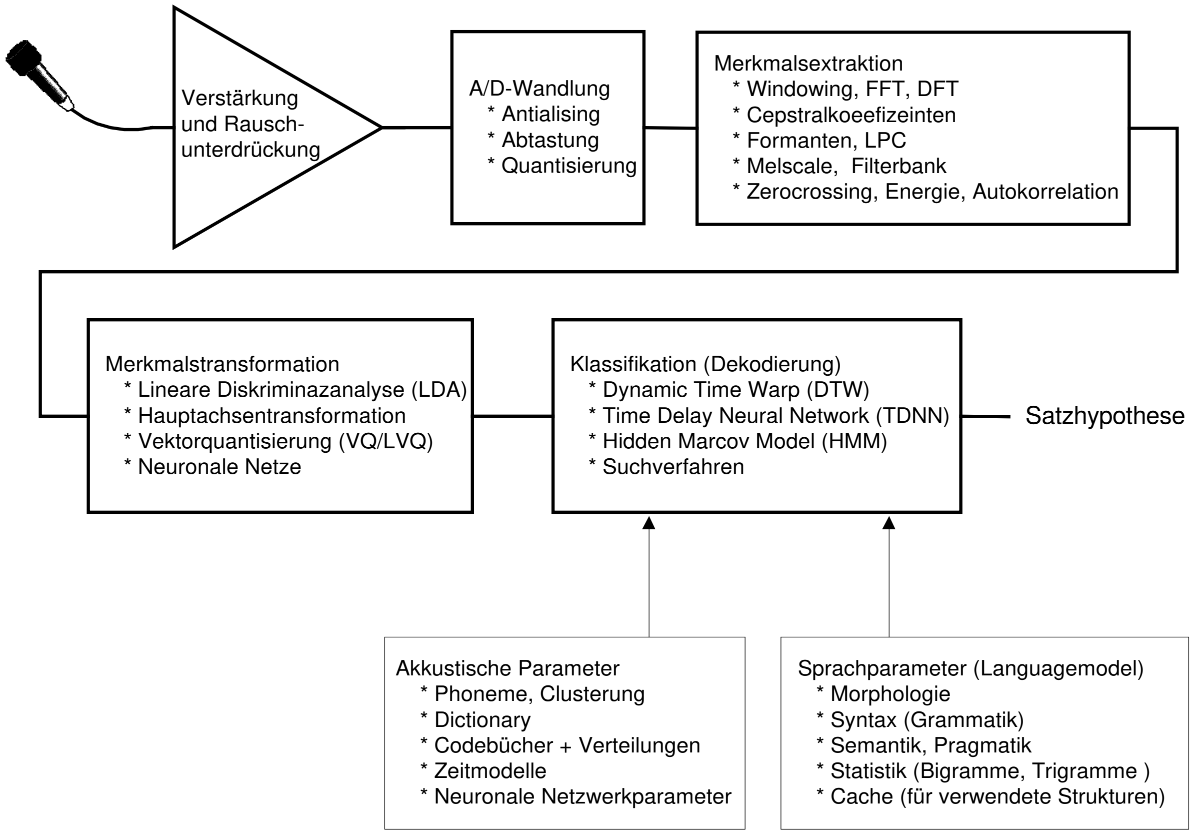

Verstärkung und Rausch-unterdrückung
A/D-Wandlung
* Antialising
* Abtastung
* Quantisierung
Merkmalsextraktion
* Windowing, FFT, DFT
* Cepstralkoeefizeinten
* Formanten, LPC
* Melscale, Filterbank
* Zerocrossing, Energie, Autokorrelation
Merkmalstransformation
* Lineare Diskriminazanalyse (LDA)
* Hauptachsentransformation
* Vektorquantisierung (VQ/LVQ)
* Neuronale Netze
Klassifikation (Dekodierung)
* Dynamic Time Warp (DTW)
* Time Delay Neural Network (TDNN)
* Hidden Marcov Model (HMM)
* Suchverfahren
Satzhypothese
Akkustische Parameter
* Phoneme, Clusterung
* Dictionary
* Codebücher + Verteilungen
* Zeitmodelle
* Neuronale Netzwerkparameter
Sprachparameter (Languagemodel)
* Morphologie
* Syntax (Grammatik)
* Semantik, Pragmatik
* Statistik (Bigramme, Trigramme )
* Cache (für verwendete Strukturen)

Abbildung 2.5: Grundmodell eines Spracherkenners

### 2.2.1 Vorverarbeitung und Digitalisierung

Zur Aufnahme der Schallwellen dient meist ein qualitativ hochwertiges Mikrofon, das oft als Headset ausgeführt ist, um immer einen gleichbleibenden Abstand zum Mund des Sprecher zu gewährleisten. Oftmals besitzt das Mikrofon schon eine integrierte Verstärkung und Rauschunterdrückung.

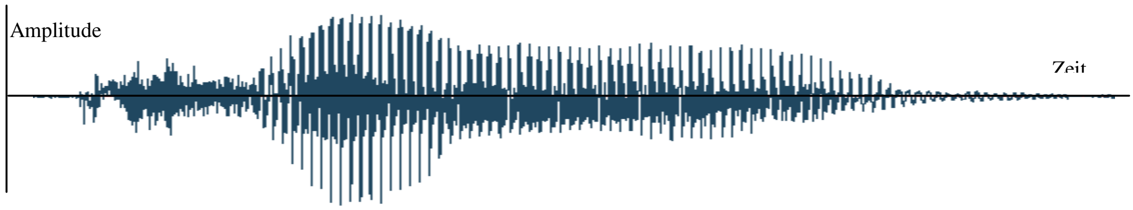


Abbildung 2.6: Analoges Signal der Äußerung “ta”

Das so erhaltene analoge Signal wird mit einem Tiefpaß gefiltert, um bei der Abtastung Antialisingeffekte zu verhindern. Nach einer Abtastung mit 16000 Hertz und einer Quantisierung auf 16 Bit erhält man ein digitales Signal mit einer Datenrate von 32 Kilobyte pro Sekunde.

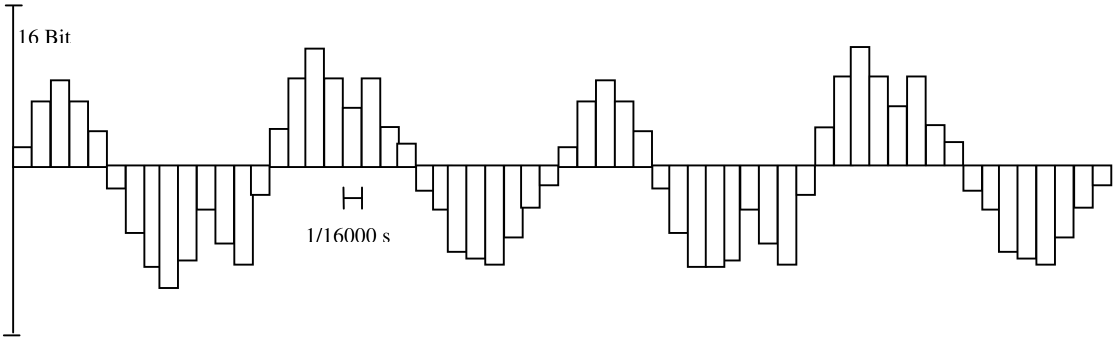


Abbildung 2.7: Digitales Signal der Äußerung “a1”

Hintergrundgeräusche können jetzt mittels digitaler Filter minimiert werden, indem in Sprachpausen das Hintergrundgeräusch analysiert wird und die Filterparameter aktualisiert werden. Das so bereinigte digitale Signal besteht aus einer ungeheuren Menge von Daten, aus denen die wichtigsten Merkmale extrahiert werden müssen. Als weniger wichtige Informationen gelten dabei die Gesamtlautstärke und die Grundfrequenz des Sprachsignals (Pitch), so daß das Sprachsignal in Bezug auf diese Größen normiert werden kann.

### 2.2.2 Merkmalsextraktion

Desweiteren wird das Sprachsignal für kurze Zeiträume (~10ms) als stationär (nicht veränderlich) angesehen. Dies macht man sich zunutze und transformiert das Signal für jedes dieser Zeitfenster (Frames) vom Zeit- in den Frequenzbereich. Dabei werden zum Herausschneiden der Zeitfenster aus dem Sprachsignal keine wirklichen Rechteckfuntionen verwendet, sondern Funktionen mit einer endlichen Bandbreite wie z.B. Hamming-, Hanning- oder Gauß-Funktionen. Im Frequenzbereich erhält man für jeden Frame sowohl ein Phasen- als auch ein Leistungsspektrum. Das Phasenspektrum erweist sich als unbedeutend, da das menschliche Gehör gegenüber Phasenverschiebungen tolerant ist, was man sich schon seit Jahrzehnten in der Telefonie zur Dämpfungskompensation zunutze macht. Somit bleibt das Leistungsspektrum übrig, welches alle wichtigen Informationen enthält.

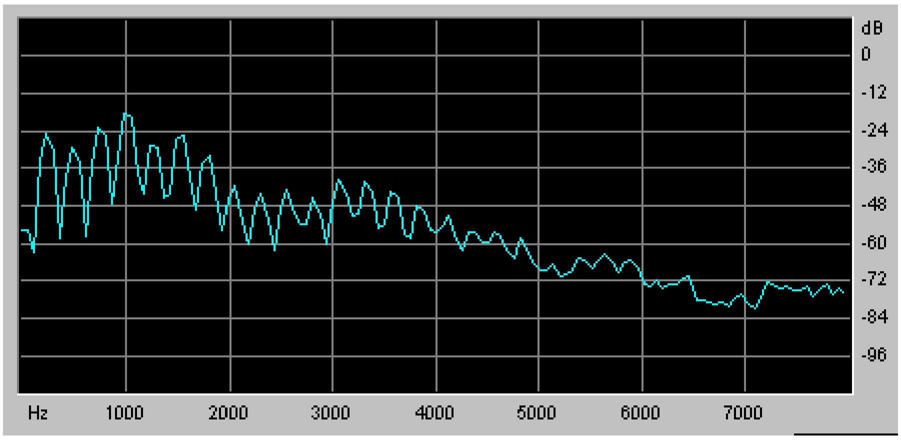


Abbildung 2.8: Leistungsspektrum der Äußerung “a1”

Wenn man für jeden Frame das Leistungsspektrum vertikal zur Zeitachse aufträgt, und dabei den Leistungswert als Grauwert darstellt (Helligkeit ~ Leistung) erhält man das Spektrogramm, welches das Sprachsignal anschaulich darstellt. Besonders einfach können hier stimmlose von stimmhaften Lauten unterschieden werden. Manche Wissenschaftler haben im Lesen von Spektrogrammen solch eine Perfektion erlangt, daß sie auf die gesprochene Äußerung schließen können.

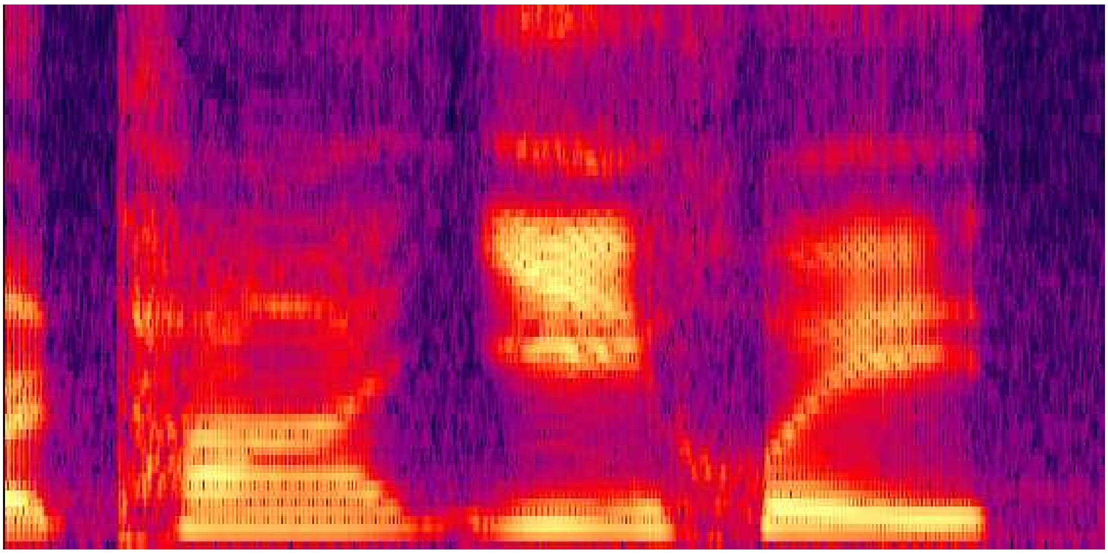
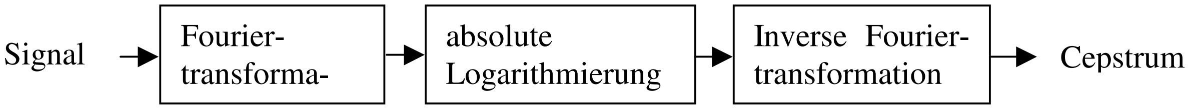


Abbildung 2.9: Spektrogramm der Äußerung “ta1 yi4 hui4”

Um die Datenrate weiter zu reduzieren, wird das Spektrum entlang der Frequenzachse quantisiert. Dies geschieht, indem man die Frequenzauflösung des menschlichen Gehörs berücksichtigt und die Frequenzbänder mit einer Breite von jeweils ca. 1 bark (= 100 mel) wählt. So erhält man z.B. pro Frame 24 mel-Spekrumkoeffizienten und liegt nun bei einer Datenrate von 24 * 100 = 2400 Werten/s. Um die Vokaltrakttransferfunktion von der Anregungsfunktion zu trennen, und somit die wichtigsten Parameter zu extrahieren, wird das Cepstrum (ein homomorpher Filter) des Signals gebildet:



Sei x(n) die Vokaltrakttransferfunktion und h(n) die Anregungsfunktion. Dann gilt:

$$x(n) * h(n) \rightarrow X(\omega) \bullet H(\omega) \rightarrow \log |X(\omega)| + \log |H(\omega)| \rightarrow FT^{-1}(\log |X(\omega)|) + FT^{-1}(\log |H(\omega)|)$$

Mittels eines Tiefpasses kann der additive Anteil der Anregungsfunktion abgeschnitten werden. Die wichtigen Parameter, die die Vokaltrakttransferfunktion beschreiben, liegen hingegen nach absteigender Relevanz sortiert im unteren Quefrequenzbereich. Die so erzielten Merkmalsvektoren können noch um andere Merkmale wie z.B.

- Merkmale aus Nachbarframes
- Deltas und Delta-Deltas gebildet aus benachbarten Frames
- Zerocrossing, Autokorrelation, Frame-Power

erweitert werden.

Die Anzahl und Auswahl der Merkmale ist dabei eher nebensächlich, da mit Methoden der Merkmalstransformation automatisch nur die wichtigsten ausgewählt und weiterverwendet werden.

### 2.2.3 Merkmalstransformation

Die Idee der Merkmalstransformation ist die, daß Merkmale eines n-dimensionalen Raums auf Merkmale eines Raums mit niedriger Dimension m ($m<n$) abbildet werden, ohne daß dabei wesentliche Information verloren geht. Alle Verfahren beruhen darauf, daß durch eine große Menge von Trainingsdaten die Transformationsparameter gelernt werden können.
Bei der Hauptachsentransformation werden die m für die Diskrimination wichtigsten Richtungen bestimmt, die dann das neue Koordinatensystem bilden. Die Hauptachsentransformation maximiert somit die Gesamtkovarianz und erleichtert infolgedessen die Trennbarkeit. Zum Berechnen der Transformation existieren effiziente, iterative Verfahren.
Bei der linearen Diskriminanzanalyse wird nicht nur die Gesamtkovarianz maximiert, sondern gleichzeitig auch die klasseninternen Kovarianzen minimiert und die Kovarianz der Mittelwerte der Klassen maximiert. Man erhält durch simultane Diagonalisierung eine ($n \times m$) dimensionale Transformationsmatrix, die durch einfache Matrixmultiplikation einen Merkmalsvektor vom n-dim. in den m-dim. Raum überführt [67].
Eine Merkmalsreduzierung kann auch durch die Verwendung neuronaler Netze erzielt werden, indem ein Netz mit n Eingangsknoten und m Ausgangsknoten auf der Trainingsmenge eingelernt wird. Später erwartet man, daß das Netz auch ungesehene Merkmale ohne wesentlichen Informationsverlust in den niederdimensionalen Raum transferieren kann [72].
Die Vektorquantisierung (VQ) reduziert den Merkmalsraum nicht durch Verminderung der Dimension, sondern teilt den Merkmalsraum in eine endliche Anzahl von Gebieten auf, so daß der Wertebereich eingeschränkt wird. Die k-nearest neighbour Estimation oder das Parzen-Window sind zwei Verfahren, die unter direkter Verwendung der Trainingsdaten die Wahrscheinlichkeit der Klassenzugehörigkeit für einen Merkmalsvektor schätzen. Problematisch bei diesen Verfahren ist die Schätzung der Systemparameter (k, Volumen V) und die Anzahl und die Auswahl der zu verwendenden Trainingsdaten, die für eine hohe Auflösung natürlich groß sein sollte, womit aber wiederum die Reduktion nur gering ausfällt. Die lernende Vektorquantisierung umgeht dieses Problem dadurch, daß für jede Klasse nur ein (oder wenige) Refe-renzvektoren gespeichert werden, die folgendermaßen gelernt werden: Bis ein gewisses Konvergenzkriterium erfüllt wird, wird jeweils ein Trainingsvektor ausgewählt, und der zu ihm nächste Klassenreferenzvektor bestimmt. Gehören nun beide Vektoren zur selben Klasse, wird der Klassenreferenzvektor leicht zum Trainingsvektor hin verschoben, sonst wird er von ihm weggeschoben. Zu diesem Basisverfahren existieren ein große Menge von Abwandlungen, die in verschiedenen Situationen Vorteile aufweisen.

### 2.2.4 Klassifikation

Nachdem die Merkmalsvektoren bestimmt und entsprechend aufbereitet wurden, muß für jeden die Wahrscheinlichkeitverteilung über die Klassen bestimmt werden, um dann mit einem Suchverfahren die erlaubte Klassenfolge zu finden, die am wahrscheinlichsten der Äußerung entspricht.

Man versucht also, unter der Annahme, daß die Äußerung A erfolgte, die wahrscheinlichste Wortfolge W zu finden. D.h. man bestimmt: $\arg\max_{W} P(W \mid A)$.

Unter Anwendung der Bayesregel kann dies folgendermaßen umgeformt werden:

$$\arg\max_{W} P(W \mid A) = \arg\max_{W} \frac{P(W) \cdot P(A \mid W)}{p(A)} \cong \arg\max_{W} P(W) \cdot P(A \mid W)$$

Mit: P(W|A) = A posteriori Wahrscheinlichkeit
P(W) = A priori Wahrscheinlichkeit
P(A|W) = Klassenbedingte Wahrscheinlichkeit
p(A) = Wahrscheinlichkeit der Äußerung

Die Maximierung der Wortfolge W hängt danach also von der Maximierung der akustischen Modellierung P(A|W) und der sprachlichen Modellierung P(W) ab.

### 2.2.4.1 Akustische Modellierung

Zur Lösung dieser Aufgabe haben sich vier unterschiedliche Hauptrichtungen herausgebildet, die durch Betonung verschiedener, teils konkurrierender Paradigmen versuchen, das komplexe Problem der automatische Spracherkennung zu bewältigen [63]:

1. Der schablonenbasierte Ansatz:
   Dieser Ansatz ist der älteste. Bei ihm werden alle wichtigen Merkmale der zu erkennenden Worte in Schablonen festgehalten. Eine gesprochene Äußerung wird dann aufgrund eines Abstandsmaß, mit allen Schablonen verglichen und die Schablonen mit dem geringsten Abstand wird als beste Worthypothese ausgewählt. Um Stauchungen auf der Zeitachse tole-rieren zu können, wird ein Verfahren aus der dynamischen Programmierung eingesetzt. Beim Dynamic Time Warp (DTW) wird eine Matrix aus den Merkmalsvektoren der Äußerung und den Vektoren der Referenzäußerung aufgebaut, deren Elemente jeweils die Distanz zwischen den beiden Vektoren enthält. Als Abstandmaß kommen dabei Maße wie die gewichtete Euklidische Distanz, die Hamming-Distanz oder die Mahalanobis-Distanz in Betracht. Aufgabe des DTW ist es nun einen Zuordnungspfad mit minimaler Abstandssumme durch die Matrix zu finden.

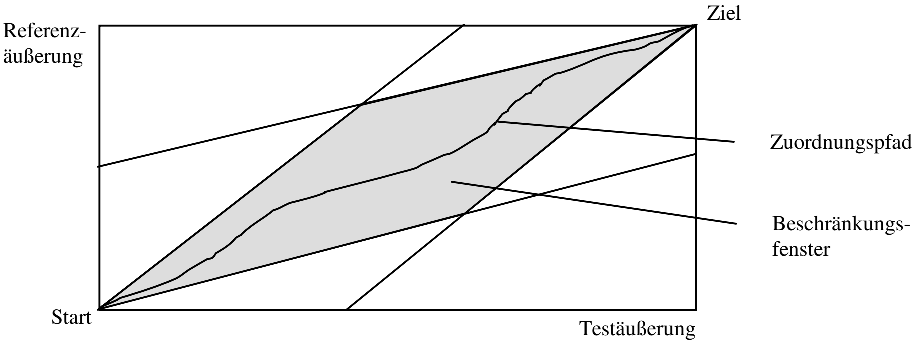


Abbildung 2.10: Dynamic Time Warp

Um den Rechenaufwand zu minimieren, wird die Menge der mögliche Zuordnungspfade auf ein Fenster beschränkt bzw. eine spezielle Warpingfunktion (z.B. Itakura, Sakoe, ...) gewählt, die die Steigung des Zuordnungspfads beschränkt. Trotzdem ist der Rechenaufwand noch sehr hoch, da die Testäußerung mit jeder Referenzäußerung auf diesem Wege verglichen werden muß. Aus diesem Grund lassen sich effektiv nur Einzelworterkenner mit kleinem Vokabular realisieren, die dann aber wiederum eine Wortendedetektion erfordern – wenn auch Ansätze zur Satzhypothesenbildung erprobt wurden [34]. Eine weitere Schwierigkeit ist, daß die Referenzäußerung speziell auf einen Sprecher zugeschnitten ist. Dies hat zur Folge, daß sich verschiedene Worte dieses Sprechers unter Umständen mehr ähneln, als das gleiche Wort von verschiedenen Sprechern. Diese Problematik läßt sich dadurch abmildern, daß für jedes Äußerung mehrere Schablonen von verschiedenen Sprecher gespeichert werden.

2. Der wissensbasierte Ansatz:
   Dieser Ansatz integriert unterschiedliche Wissensquellen in einen Erkennungsprozeß. Dabei kommen als Wissensquellen in Betracht:
   - Phonetik: Wissen über die Charakteristik der Sprachgeräusche
   - Phonologie: Variabilität der Aussprache
   - Prosodie: Rhythmus und Betonung der Sprache
   - Lexikon & Morphologie: Wissen über Worte und deren Bildung
   - Syntax: grammatische Regeln
   - Semantik: Wissen über die Bedeutung der Sprache
   - Pragmatik: Kontextwissen der Konversation
   - Ableitungs- und Inferenzregeln: Folgerungen aufgrund externen Wissens

   Die einzelnen Wissensquellen sind allerdings während des Erkennungsprozesses keineswegs unabhängig voneinander. Sie werden meistens aufgrund ihrer Komplexität einzeln betrachtet und immer wieder müssen Schlußfolgerungen einer Quelle aufgrund neuer Erkenntnisse einer anderen revidiert werden [49]. Der wissensbasierte Ansatz wird heute nicht mehr alleine genutzt, sondern ist eine Möglichkeit andere Ansätze zu bereichern.

3. Der konnektionistische Ansatz:
   Während der Renaissance der neuronalen Netze Ende der 80‘er Jahre, wurden auch viele konnektionistische Ansätze in der Spracherkennung erforscht. Besondere Beachtung haben dabei zwei Vorgehensweisen gefunden: Zum einen die Kombination mit statistischen Verfahren, sogenannte hybride Ansätze [32][33], bei denen neuronale Netze zur Schätzung der Ausgabewahrscheinlichkeiten dienen und zum anderen Time Delay Neural Networks (TDNN), die speziell für die Spracherkennung entwickelt wurden [63], so daß sie auch zeitveränderliche Muster unabhängig von der zeitlichen Verschiebung und Länge erkennen können.

   TDNNs sind neuronale Netzwerke mit einer speziellen, geschichteten Topologie, bei denen die Aktivierung eines Neurons nicht einfach durch die gewichtete Summe der Eingabe abhängt, sondern auch von weiter zurückliegenden Eingaben. Als Schwellwertfunktion wird eine sigmoide Funktion benutzt, um die nötige Differenzierbarkeit für ein abgewandeltes Backpropagationverfahren zu gewährleisten. In der Lernphase dieses speziellen Backpropagationalgorithmus werden die zusammengehörigen, zeitversetzten Gewichte als gekoppelt betrachtet (weight sharing) und alle auf den gemeinsamen, gelernten Mittelwert gesetzt. Solche Netzwerke sind in der Lage, zwischen sonst nur schwer unterscheidbaren Konsonanten, wie z.B. ‘bdg’ zu differenzieren [62]. Allerdings steigt mit der Anzahl der Spracheinheiten auch die Netzgröße und damit auch die benötigten Trainingsdaten sprunghaft an, so daß Verfahren für eine bessere Skalierbarkeit gesucht wurden. Diese wurden in den hierarchischen TDNNs gefunden, bei denen mehrere schon trainierte Netze zusammengefügt wurden, und nur noch ein Verbindungsnetz (connectionist clue) trainiert wird. So erreicht man eine nahezu lineare Skalierbarkeit in Bezug auf die zu erkennenden Spracheinheiten. Um auch längere Spracheinheiten als Phoneme erkennen zu können, wird das Konzept zu den Multi-State TDNNs erweitert. Diese Erweiterung besteht im wesentlichen aus einer weiteren Schicht, die zwischen Phonemschicht und Ausgabeschicht eingefügt wird, um dort Phonemfolgen zu repräsentieren, so daß dort eine Art Dynamic Time Warp erfolgt.

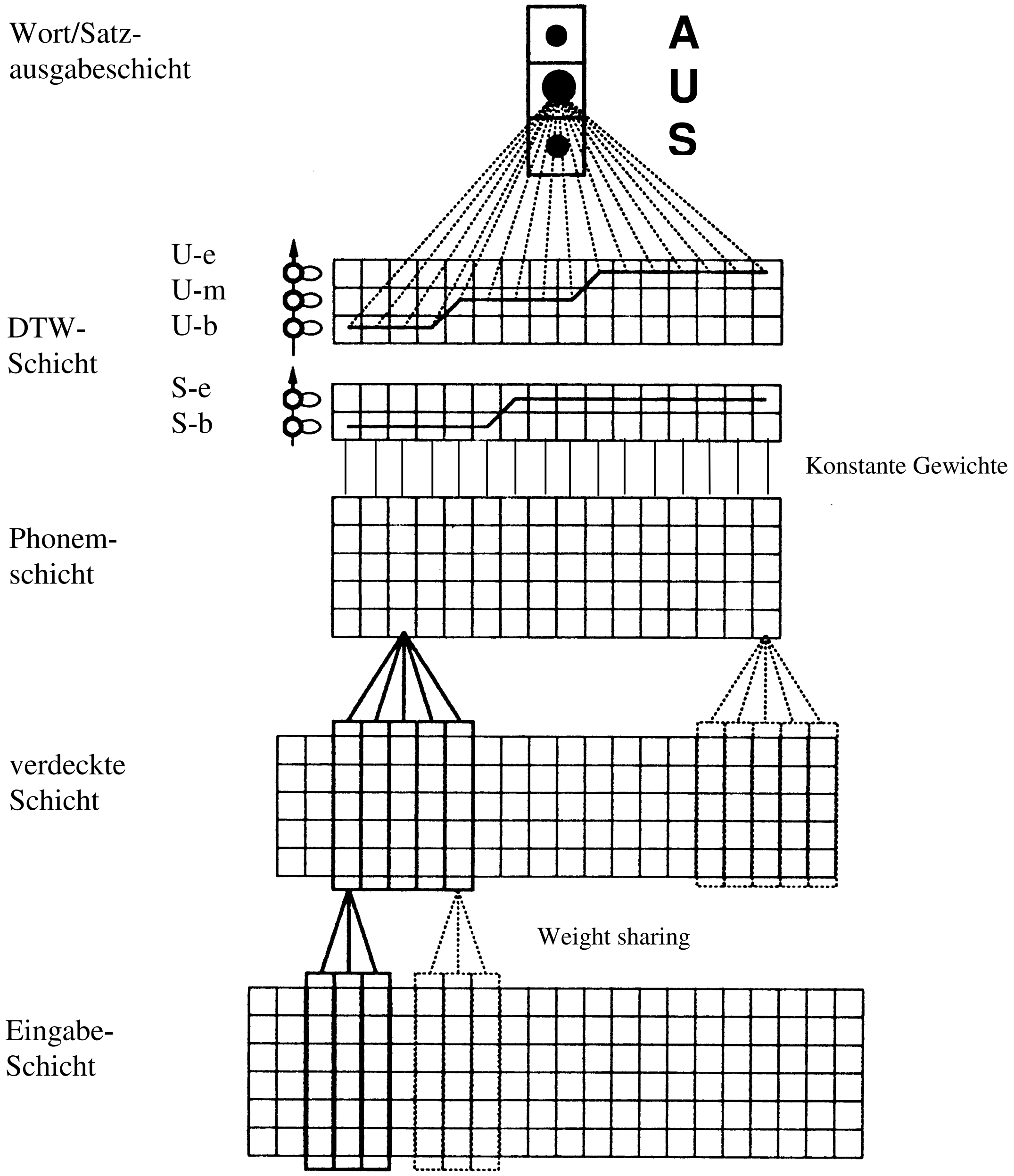

Wort/Satz-
ausgabeschicht
A
U
S
U-e
U-m
U-b
DTW-
Schicht
S-e
S-b
Konstante Gewichte
Phonem-
schicht
verdeckte
Schicht
Weight sharing
Eingabe-
Schicht

Abbildung 2.11: Architektur des MS-TDNN

Als Vorteil dieses Ansatzes kann die Gleichförmigkeit angesehen werden (d.h. für alle Teile werden nur NNs eingesetzt). Es sind keine einschränkenden Annahmen erforderlich. Allerdings ist im allgemeinen ein teures Training unvermeidlich.

4. Der stochastische Ansatz:
   Im Laufe der Zeit hat der stochastische Ansatz alle anderen Ansätze mehr oder weniger verdrängt, ihnen Nischenplätze zugewiesen oder sie als Teil integriert. Besonders die Hidden Marcov Modelle (HMMs) haben sich als unverzichtbar erwiesen. Sie sind in allen aktuellen Erkennern zu finden.
   Das Hidden Marcov Modell benutzt einen zweistufigen stochastischen Prozeß zur Beschreibung oder Simulation einer Signalquelle. Es wird definiert durch:

| | |
|---|---|
| Zustandsmenge | $S = \{S_1, ..., S_n\}$ |
| Ausgabemenge | $V = \{V_1, ..., V_m\}$ |
| Zustandsübergangswahrscheinlichkeiten | $A = \{a_{ij} \mid a_{ij} = P(q_{t+1} = S_j \mid q_t = S_i),\ 1 \leq i,j \leq n\}$ |
| Ausgabewahrscheinlichkeitsverteilungen | $B = \{b_j(k) \mid b_j(k) = P(V_k \mid q_t = S_J),\ 1 \leq j \leq n;\ 1 \leq j \leq m\}$ |
| Anfangswahrscheinlichkeiten | $\pi = \{\pi_i \mid \pi_i = P(q_1 = S_i),\ 1 \leq i \leq n\}$ |

Die zwei Stufen des Zufallsprozesses sind die Ausgabewahrscheinlichkeitsverteilungen und die Zustandsübergangswahrscheinlichkeiten der Hintergrundsequenz, die dem Betrachter vollständig verborgen bleibt, da er nur die Ausgaben des Prozesses sieht. Für die Sprach-erkennung werden vorwärtsgerichtete HMMs verwendet, in denen nur Selbstzyklen erlaubt sind. Zeitliche Verzerrungen werden durch Selbstzyklen oder Abkürzungen auf der Zustands folge mittels der Zustandsübergangswahrscheinlichkeiten modelliert, Aussprachevarianten durch die Ausgabewahrscheinlichkeitsverteilungen. Unter Annahme der Markov-Annahmen, daß erstens ein Zustand nur von seinem Vorgänger abhängig ist und zweitens die Ausgabe nur vom aktuellen Zustand abhängig ist, können Worte oder Phrasen mit HMMs modelliert werden. Es stellen sich zum Einsatz in einem Spracherkenner folgenden Fragen:

a) **Evaluation**: Wie wahrscheinlich ist eine Äußerung (Beobachtungssequenz) für ein gegebenes HMM ?
b) **Dekodierung**: Was ist die optimale Zustandsfolge für eine gegebene Äußerung (Beobachtungssequenz) ?
c) **Training**: Wie bestimmt man die Parameter $\lambda = (A,B, \pi)$ eines HMMs zur Modellierung eines Wortes ?
d) **HMM-Typ:** Was ist die optimale Topologie, die richtige Menge von Zuständen und das beste Ausgabealphabet ?

Zu alle diesen Problemen gibt es befriedigende Anworten, was HMMs zum idealen Werkzeug in der Spracherkennung (und anderen Gebieten) macht. Im folgenden werden diese Antworten kurz skizziert:

a) **Evaluation**:
   Geg.: Beobachtungssequenz $O = O_1, ..., O_r$, HMM $\lambda = (A,B, \pi)$
   Ges.: Wahrscheinlichkeit der Beobachtungssequenz $P(O \mid \lambda)$

$$P(O \mid \lambda) = \sum_{alleQ} P(O \mid Q, \lambda) \cdot P(Q \mid \lambda) = \sum_{q_1 \ldots q_r} \left(\pi_{q_1} \cdot b_{q_1}(O_1)\right) \cdot \left(a_{q_1 q_2} \cdot b_{q_2}(O_2)\right) \cdot \ldots \cdot \left(a_{q_{r-1} q_r} \cdot b_{q_r}(O_r)\right)$$

   Der obige direkte Ansatz summiert über alle möglichen Zustandsfolgen auf, was aber zu einem exponentialen Aufwand führt und somit nicht in Betracht kommen kann. Der Forward-Backward-Algorithmus setzt auf dynamisches Programmieren und verringert den Aufwand auf $O(T \cdot N^2)$, indem er für jeden Zustand, von einem Zeitpunkt zum nächsten iterativ die Wahrscheinlichkeiten aufaddiert. Dazu werden die Zustände des HMMs auf der Zeitachse ausgerollt, so daß eine Matrix aus Zuständen und Zeitpunkten entsteht.

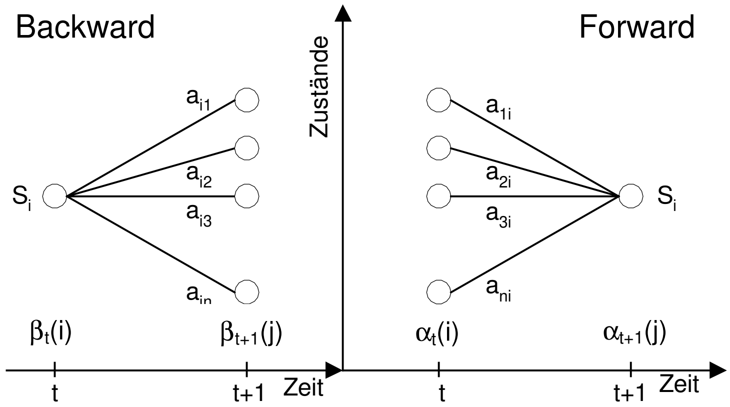


Abbildung 2.12: Forward-Backward-Trellis

Die Forward-Variablen $\alpha_t(i)$ bzw. die Backward-Variablen $\beta_t(i)$ enthalten für einem Zustand die bis zu Zeitpunkt t aufaddierten Wahrscheinlichkeiten für alle Pfade, die sich rekursiv aus den vorherigen $\alpha_{t-1}(i)$ bzw. $\beta_{t-1}(i)$ ergeben. Für die Forward-Variable gilt:

$$\begin{array}{ll} \text{Def.:} & \alpha_t(i) := P(O_1 \ldots O_t, q_t = S_i \mid \lambda) \\ \text{Rekursion:} & \alpha_1(i) = \pi_i b_i(O_1) \qquad \text{i} = 1\ldots\text{n} \\ & \alpha_{t+1}(i) = \left( \sum_{j=1}^{n} \alpha_t(j) a_{ji} \right) \cdot b_i(O_{t+1}) \quad \text{t} = 1\ldots\text{T - 1, i} = 1\ldots\text{n} \end{array}$$

Um die Wahrscheinlichkeit für die Beobachtungssequenz O zu erhalten, müssen die

$$P(O \mid \lambda) = \sum_{j=1}^{n} \alpha_T(j)$$

Werte der Forward-Variablen am letzten Zeitpunkt T aufaddiert werden:
Für den Backward-Pfad gilt entsprechendes.

b) **Dekodierung**

Geg.: Beobachtungssequenz $O = O_1, ..., O_r$, HMM $\lambda = (A,B,\pi)$
Ges.: optimale (wahrscheinlichste) Zustandsfolge $Q = Q_1, ..., Q_r$
Der Ansatz ist hier derselbe wie bei der Evaluation. Nur tritt an die Stelle der Sumation eine Maximumsbildung:

$$\begin{array}{ll} \text{Def.:} & \delta_t(i) := \max_{q_1 \ldots q_t} \left( P(O_1 \ldots O_t, q_t = S_i \mid \lambda) \right) \\ \text{Rekursion:} & \delta_1(i) = \pi_i b_i(O_1) \qquad \text{i} = 1\ldots\text{n} \\ & \delta_{t+1}(i) = \max_j \left( \delta_t(j) a_{ji} \right) \cdot b_i(O_{t+1}) \quad \text{t} = 1\ldots\text{T - 1, i} = 1\ldots\text{n} \end{array}$$

Um die wahrscheinlichste Zustandsfolge zu erhalten, muß in einem Backtracking-Pass zur Erzeugung des Pfades immer der Knoten mit dem größten δ-Wert gewählt werden.

c) **Training der HMM-Parameter**

Die Parameter A (Zustandsübergangswahrscheinlichkeiten) und B (Ausgabewahrschein lichkeiten) werden anhand einer möglichst großen Trainingsmenge gelernt. Dabei kommt das iterative EM-Verfahren nach Baum Welch zu Einsatz. Die Anfangswahrscheinlichkeiten π können als Spezialfall der Zustandsübergangswahrscheinlichkeiten A angesehen werden, und werden deshalb nicht gesondert betrachtet.

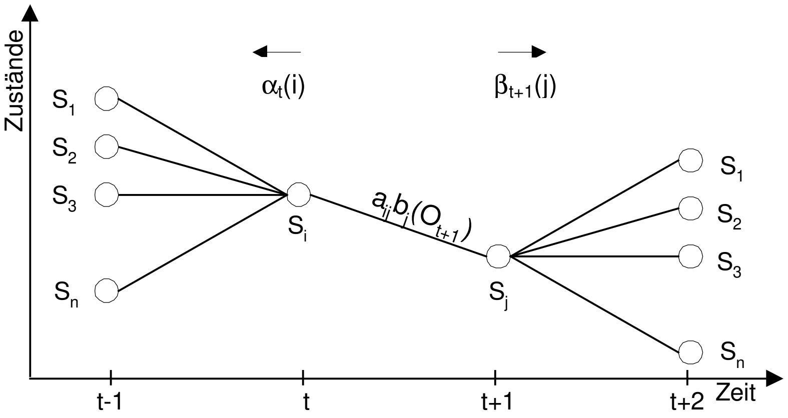


Abbildung 2.13: Forward-Backward-Reestimation

Die eingeführte Variable γ gibt die Wahrscheinlichkeit für einem Übergang von $S_i$ nach $S_j$, bei Ausgabe von $O_{t+1}$, an. Für γ gilt:

$$\gamma_t(i,j) := P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda)$$

$$\gamma_1(i,j) = \frac{P(q_t = S_i, q_{t+1} = S_j, O \mid \lambda)}{P(O \mid \lambda)}$$

$$= \frac{\alpha_t(\mathrm{i}) \cdot \mathrm{a}_{\mathrm{ij}} \mathrm{b}_{\mathrm{j}}(\mathrm{O}_{\mathrm{t+1}}) \cdot \beta_{\mathrm{t+1}}(\mathrm{j})}{\sum_{\mathrm{i=1}}^{\mathrm{n}} \sum_{\mathrm{j=1}}^{\mathrm{n}} \alpha_t(\mathrm{i}) \cdot \mathrm{a}_{\mathrm{ij}} \mathrm{b}_{\mathrm{j}}(\mathrm{O}_{\mathrm{t+1}}) \cdot \beta_{\mathrm{t+1}}(\mathrm{j})}$$

Mit Hilfe dieser Variablen γ können die Erwartungswerte für $a_{ij}$, $b_{ij}(k)$ folgendermaßen neu bestimmt werden:

$$\bar{\mathrm{a}}_{\mathrm{ij}} = \frac{\text{Erwartete Anzahl Übergänge von } S_i \text{ nach } S_j}{\text{Erwartete Anzahl Übergänge von } S_i \text{ aus}} = \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{j=1}^{n} \sum_{t=1}^{T} \gamma_t(i,j)}$$

$$\bar{\mathrm{b}}_{\mathrm{ij}}(k) = \frac{\text{Erwartete Anzahl, daß im Zustand j Symbol k ausgegeben wird}}{\text{Erwartete Anzahl überhaupt im Zustand j zu sein}} = \frac{\sum_{t:y_t=k} \sum_{i=1}^{n} \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_{i=1}^{n} \gamma_t(i,j)}$$

Mit diesen neuen Schätzwerten für $a_{ij}$, $b_{ij}(k)$ können verbesserte α,β und damit ein neues, genaueres γ berechnet werden. Es ergibt sich das folgende iterative Verfahren:

1. Initialisiere $\gamma = (A,B)$
2. Berechne $\alpha,\beta,\gamma$
3. Schätze $\gamma = (A,B)$ anhand $\gamma$
4. Ersetze $\gamma$ durch $\gamma$
5. Falls Gütekriterium erfüllt halte an, sonst weiter bei 2.

Dieses Algorithmus konvergiert zu einem lokalen Minimum, so daß bei Konvergenz und nicht ausreichender Güte, abgebrochen werden muß und $\gamma$ neu zu initialisieren ist. Die Initialisierung kann gleichverteilt, zufällig oder mittels des kmeans-Verfahren erfolgen.

d) **HMM-Typ:**
Die Frage nach dem besten HMM-Typ läßt sich nicht so einfach beantworten. Sie ist von Fall zu Fall verschieden. In der Spracherkennung haben sich Vorwärts-Topologien, die kein Rückwärtsschreiten erlauben, als sinnvoll erwiesen.

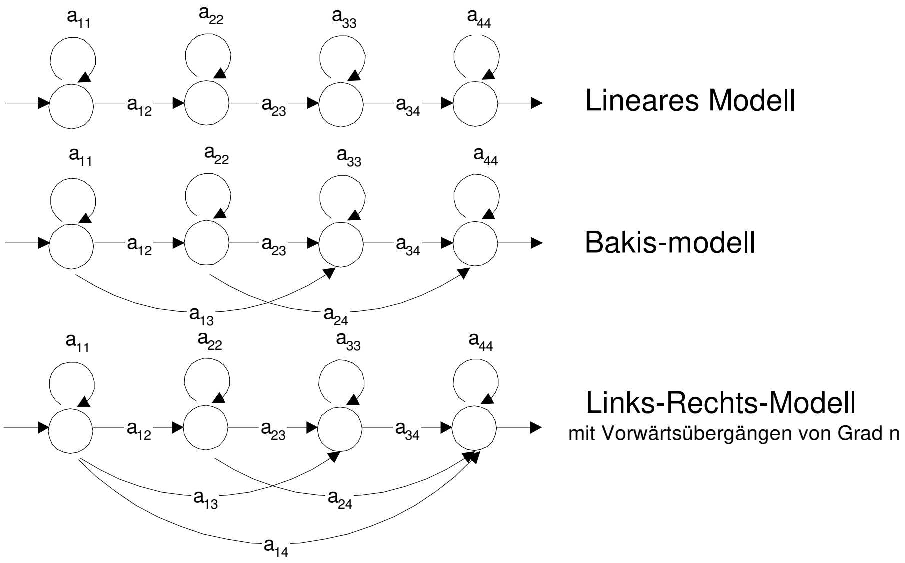

$a_{11}$
$a_{22}$
$a_{33}$
$a_{44}$
$a_{12}$
$a_{23}$
$a_{34}$
Lineares Modell
$a_{11}$
$a_{22}$
$a_{33}$
$a_{44}$
$a_{12}$
$a_{23}$
$a_{34}$
Bakis-modell
$a_{13}$
$a_{24}$
$a_{11}$
$a_{22}$
$a_{33}$
$a_{44}$
$a_{12}$
$a_{23}$
$a_{34}$
Links-Rechts-Modell
mit Vorwärtsübergängen von Grad n
$a_{13}$
$a_{24}$
$a_{14}$

Abbildung 2.14: HMM - Topologien

Weitere wichtige Entscheidungen die getroffen werden müssen sind:

- Stellen die Zustände Silben, Phoneme, Subphoneme oder andere akustische Einheiten dar?
- Sind die Ausgabeverteilungen diskret (durch Vektorquantisierung), kontinuierlich (z.B. multimodal) oder semikontinuierlich?
- Werden Parameter gekoppelt?

In HMMs können andere Wissensquellen konsistent integriert werden, indem sie Wahrscheinlichkeitswerte gewichten. Obwohl HMMs auf einer mathematischen Basis stehen, muß man immer bedenken, daß die Markov-Annahmen vorausgesetzt werden.

### 2.2.4.2 Sprachliche Modellierung

Wenn ein menschliches Gespräch durch Nebengeräusche für kurze Augenblicke überdeckt wird, Aussprachefehler oder Versprecher auftraten, ist es dem Zuhörer trotzdem möglich, die vollständige Äußerung zu rekonstruieren. Er benutzt dabei sprachliches und anderes Hintergrundwissen. Auch Spracherkenner nutzen dieses sprachliches Wissen um die Erkennungsleistung zu verbessern oder die Suche effizienter und schneller durchführen zu können.

Um die Sprache zu modellieren, können unterschiedliche Methoden angewandt werden. Allerdings haben hier wiederum die statistischen Verfahren die anderen Verfahren verdrängt, so daß kontextfreie Grammatiken, endliche Automaten und dergleichen nur noch in Systemen mit einer kleinen Spezialgrammatik, wie z.B. in Menü- oder Bediensystemen eingesetzt werden. In Diktiersystemen kommen hingegen fast ausschließlich Systeme, die auf Wortstatistiken beruhen, zum Einsatz.

Wenn die Wahrscheinlichkeit P(W), mit der eine Wortfolge gesprochen wird, bestimmt werden kann, können somit grammatikalisch falsche oder unsinnige Sätze ausgeschlossen werden und häufig benutzte Sätze hypothetisiert werden. P(W) läßt sich allerdings nicht direkt bestimmen, da hierzu eine unbegrenzte Menge von Trainingsdaten und entsprechender Rechenaufwand nötig wäre. P(W) wird daher auf einer begrenzten Trainingsmenge geschätzt.

$$P(W) = P(w_0, \ldots, w_n) = \prod_{i=1}^{n} P(w_i \mid w_0, \ldots, w_{i-1}) \cdot P(w_0)$$

Unter der Annahme, daß $w_i$ stärker von $w_k$ als von $w_{k-j}$ für $0<j<k<i$ abhängt, können Äquivalenzklassen gebildet werden, die nur noch einen beschränkten Kontext links von $w_i$ betrachten. Je nachdem wie breit der Kontext ist, spricht man von Unigrammen (Kontextbreite 0), Bigrammen (Kontextbreite 1) und Trigrammen (Kontextbreite 2).

$$\text{Unigramme:} \quad P(W) = \prod_{i=0}^{n} P(w_i)$$

$$\text{Bigramme:} \quad P(W) = P(w_0) \cdot \prod_{i=1}^{n} P(w_i \mid w_{i-1})$$

$$\text{Trigramme:} \quad P(W) = P(w_0) \cdot P(w_1) \cdot \prod_{i=2}^{n} P(w_i \mid w_{i-2} w_{i-1})$$

N-Gramme mit einem größeren Kontext kommen im allgemeinen nicht in Betracht, da die Trainingsmenge über dem Vokabular V in einer Größenordnung von $|V|^N$ liegen sollte. Bei einem vergleichsweise kleinen Vokabular von 1000 Worten ist schon bei Trigrammen ein Korpus von etwa $10^9$ Worten notwendig. Um den Mangel an Trainingsdaten zu kompensieren, werden Verfahren angewandt, die entweder die Wahrscheinlichkeiten von Trigrammen mit denen von Bigramme und Unigramme mischen (Deleted Interpolation/Glättung), oder einen Rückfall (Backoff) bei nicht robust geschätzten Trigrammen auf Bigramme und Unigramme erlauben. Dabei müssen korrekte Gewichtungsfaktoren bestimmt werden, damit die unterschiedlichen Wahrscheinlichkeiten miteinander verrechnet werden können. Eine weitere Möglichkeit für eine robustere Schätzung ist die Klassenbildung von Wortgruppen oder Wortarten. Die Schätzung der N-Gramme erfolgt über das Auszählen derselben im Trainingskorpus:

$$\text{Trigramme:} \quad P(w_k \mid w_{k-1} w_{k-2}) \cong \frac{\#(w_{k-2} w_{k-1} w_k)}{\#(w_{k-2} w_{k-1})}$$

$$\text{Bigramme:} \quad P(w_k \mid w_{k-1}) \cong \frac{\#(w_{k-1} w_k)}{\#(w_{k-1})}$$

$$\text{Unigramme:} \quad P(w_k) \cong \frac{\#(w_k)}{\#Worte}$$

### 2.2.4.3 Dekodierung / Suche

Bei Spracherkennern für kontinuierliche Sprache kann aus Gründen nicht ausreichender Trainingsdaten, mangelnden Speicherplatz und Rechenzeit nicht für jede mögliche Äußerung ein eigenes HMM trainiert werden. Infolgedessen werden Einzelwort-HMMs zu einer Gesamtstruktur zusammengefügt, in der mittels effizienten Suchverfahren die wahrscheinlichste Zustandsfolge für die Äußerung ermittelt wird. Dabei gehen neben den akustischen Informationen auch die aus der Sprachmodellierung ein.

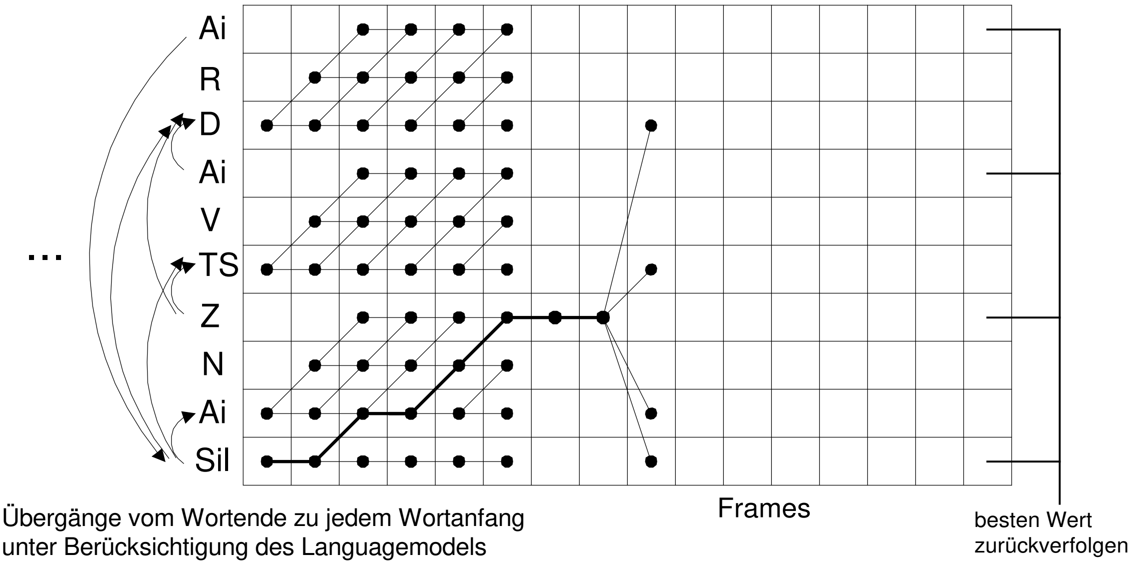

Ai
R
D
Ai
V
TS
Z
N
Ai
Sil
...
Übergänge vom Wortende zu jedem Wortanfang unter Berücksichtigung des Languagemodels
Frames
besten Wert zurückverfolgen

Abbildung 2.15: Viterbi-Suche

Die Viterbi-Suche ist eine zeitsynchrone Breitensuche, da alle erlaubten Pfade gleichzeitig verfolgt werden. Jeder Pfad besteht somit aus derselben Anzahl von Frames und kann mit anderen verglichen werden. Dies macht man sich zunutze, um aus der großen Anzahl von Pfaden weniger Erfolg versprechende zu verwerfen (Pruning). Dabei setzt man einen Wert (Beam) fest, so daß ein Pfad genau dann verworfen wird, wenn er mehr als um diesen Wert vom besten Pfad abweicht. Eine weitere Methode zur Geschwindigkeitssteigerung ist die, daß man an jedem Wortende die Anzahl der möglichen Folgeworte einschränkt (Top N). Um den Speicherplatzbedarf einigermaßen in Grenzen zu halten, werden die Strukturen dynamisch bei Bedarf erweitert und wieder abgebaut. Die Viterbidecodierung findet nicht die gewünschte $P(O|Q,\lambda)$ sondern nur die Näherung $P(O,Q|\lambda)$.

Die Kellersuche (Stack Decoding) hingegen findet die beste Wortsequenz durch eine A*-Tiefensuche und ist somit konsistent zum Forward-/Backward-Training. Um aber Pfade aus der Suche auszuschließen, müssen verschieden lange Pfade miteinander verglichen werden. Dies führt zu einer aufwendigen, problematischen Längennormalisierung und führt mit dem Aufwand der Kellerverwaltung zu Geschwindigkeitsnachteilen gegenüber der Viterbi-Suche.

# 3 Die chinesische Sprache

Da die chinesische Sprache sich grundlegend von den westlichen Sprachen unterscheidet, ist es notwendig deren Eigenschaften und Besonderheiten näher zu beleuchten, um Entwurfsentscheidungen der vorliegenden Arbeit nachvollziehen zu können. Eine praktische Einführung in die chinesische Sprache geben die Bücher [39] und [40].

Zum Verständnis der chinesischen Sprache ist deren geschichtliche Entwicklung besonders wichtig. Obwohl die gesprochene Sprache vor der Schrift existierte, läßt sich über Lautgebung vor der Entwicklung der Schrift nur sehr wenig sagen. Es ist ja gerade die Schrift, die eine Weitergabe von Informationen über viele Generationen hinweg vereinfacht.

In den frühen Epochen war die Entwicklung der Schrift mit der Entwicklung der chinesischen Zeichen nahezu identisch. Somit kommt der Entwicklung der Zeichen eine besondere Bedeutung zu.

## 3.1 Entwicklung der Zeichen

Die Triebfeder für die Entwicklung einer Schrift war die Notwendigkeit, wie in vielen anderen Kulturen sicher auch, sich angeeignetes Wissen weiterzugeben. Die chinesische Schrift begann mit bildlichen Aufzeichnungen. Die ersten bekannten Aufzeichnungen waren Abbildungen des täglichen Lebens (oft als Höhlenmalerei noch erhalten). Diese sind aber noch keine wirkliche Schrift, sondern nur eine Vorstufe, denn sie konnten weder durch gesprochene Sprache exakt wiedergegeben werden, noch konnten sie alle Bestandteile der gesprochenen Sprache wiedergeben.

Nach chinesischen Überlieferungen soll ca. 3000 Jahre vor Chr. erstmals der Beamte „Cang Jie“ am Hofe des Kaiser „Huang Di (Gelber Kaiser)“ damit begonnen haben, Symbolaufzeichnungen zu systematisieren und damit den Grundstein für ein Schriftsystem zu legen [4]. Die Überlieferungen enthalten viele mythische Elemente. So soll Cang Jie z.B. vier Augen besessen haben. Es ist somit nicht einfach festzustellen, inwieweit die Überlieferungen einen Wahrheitsgehalt haben, wie wohl die Existenz des Beamten „Cang Jie“ doch als sehr wahrscheinlich gilt.

Der älteste chinesische Symbolschriftfund ist wahrscheinlich der in Ton eingeritzte Fund im Dorf Banpo nahe bei Xi`an (s. [4]). Der Fund von 1972 wird auf nahezu 6000 Jahre datiert.

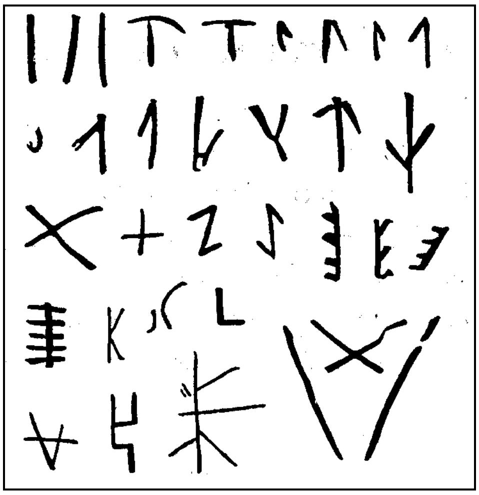


Abbildung 3.1: Erste Schriftfunde auf Tonscherben in Banpo

Diese eingeritzten Symbole haben alle noch eine sehr einfache Struktur, kennzeichnen aber den Beginn der Formung der ersten chinesischen Zeichen. Denn sie haben nicht mehr bloß einen dekorativen Charakter oder stellen eine reine Abbildung dar, sondern lassen eine erste Systematik erkennen. Andere Funde brachten Aufzeichnungen auf Knochen, Schildkrötenpanzern und Steinen zutage.
Ein anderes im frühen China entwickeltes Aufzeichnungssystem benutzte Knoten verschiedener Größe und Farbe an Seilen und Schnüren. Dieses eignete sich besonders zum Zählen, für verwaltungstechnische Angelegenheiten und als Kalender für Ereignisse und dergleichen (ein Vergleich zum berühmten Knoten im Taschentuch zwingt sich einem auf).

Für die weitere Entwicklung der chinesischen Schrift lassen sich sechs Kategorien der Zeichenkonstruktion verfolgen (nach Xu Shen):

1. Kategorie der reinen Piktogramme:

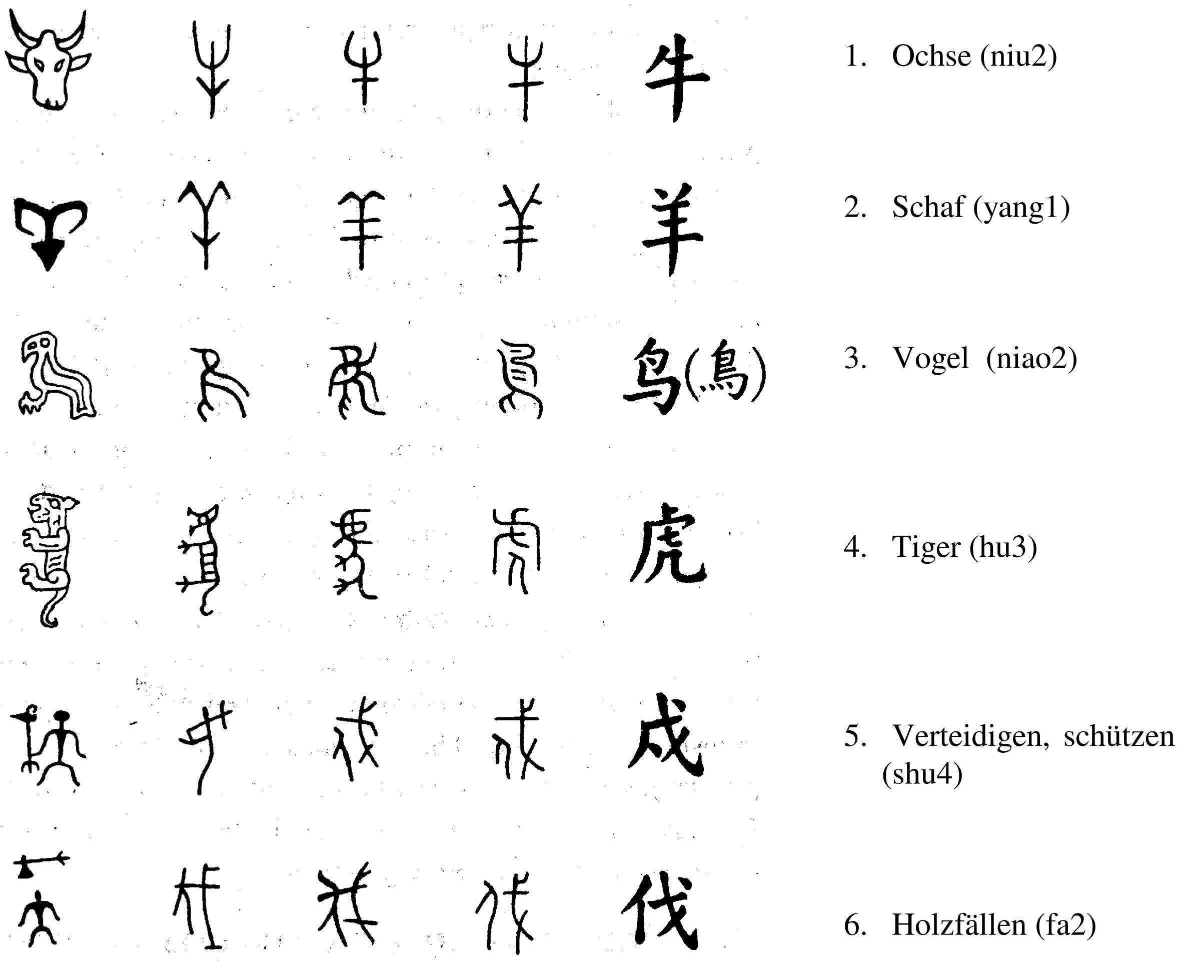


Abbildung 3.2: Entwicklung von der frühen Bilderschrift zu heutigen chinesischen Zeichen

Beim Piktogramm läßt sich eine Linie von der rein bildlichen Darstellung (links) bis zum heutigen Zeichen (rechts) verfolgen. Oftmals ist es somit in dieser Kategorie möglich, von dem Aussehen des Zeichens auf seine Bedeutung zu schließen, wenn auch sicher ein gehöriges Maß von Phantasie und Erfahrung dafür nötig ist. Fast alle Zeichen dieser Kategorie stammen aus der ersten Periode der Sprachentwicklung. Nur wenige kamen später hinzu (wie z.B. 伞 Regenschirm, yu3; 凸 konvex, tu1; 凹 konkav, ao1). Alle diese Zeichen haben gemeinsam, daß sie nicht weiter in untergeordnete Sinneinheiten unterteilt werden können, aber als Bildungselement für neue Zeichen dienen können. Nur ca. 4% der modernen chinesischen Zeichen fallen in diese Kategorie.

2. Kategorie der indikatorischen Zeichen:
   Diese Zeichengruppe benutzt Symbole, um einen abstrakten Sachverhalt auszudrücken (z.B. 上 oben, 下 unten, 一 eins, 二 zwei, 三 drei). Oft werden diese Symbole mit Zeichen der ersten Kategorie überlagert. Dabei werden die Zeichen, ähnlich wie zwei Folien, übereinandergelegt. (z.B. 木 Piktogramm für Baum + 一 Symbol für eins = 本 Wurzel oder 末 Spitze, Ende). Auch diese Zeichen können nicht weiter in untergeordnete Sinneinheiten unterteilt werden. Sie bilden die kleinste Gruppe mit unter 1% aller Zeichen.

3. Kategorie der assoziativen Zeichen:
   In dieser Gruppe sind Zeichen enthalten, die aus denen der vorherigen Gruppen kombiniert werden. Dabei werden die Zeichen verkleinert und neben- bzw. übereinander gruppiert, ohne daß Striche der einzelnen Zeichen sich dabei kreuzen. Die Beziehung zwischen den einzelnen Elementen ergibt hierbei dann die neue Bedeutung. Beispiele:
   Piktogramm Mensch 人 + Piktogramm Mensch 人 = 从 folgen (Mensch folgt Mensch)
   Piktogramm Baum 木 + Piktogramm Baum 木 + Piktogramm Baum 木 = 森 Wald
   Piktogramm Mensch 人 + Piktogramm Baum 木 = 休 ausruhen (Mensch lehnt an Baum)
   Diese Gruppe macht ca. 13 % der chinesischen Zeichen aus.

4. Kategorie der Piktogramm-Phonetischen Zeichen:
   Die dritte Kategorie erlaubt einen größeren Wortschatz zu bilden. Aber oft führt die reine Assoziation zu sehr komplexen Strukturen, die aus vielen einzelnen Komponenten zusammengesetzt sind. Durch die Abkehr von der reinen Bedeutungsorientierung können sehr einfach neue Zeichen geschaffen werden. Bestehende Zeichen werden um eine phonetische Einheit ergänzt. Beispielsweise ergibt Piktogramm Mund 口 + phonetische Einheit 土 (tu3) die neue Piktogramm 吐 (er)brechen (Bedeutungseinheit Mund, Aussprache tu3).
   Leider haben die meisten Zeichen im Laufe der Zeit ihre phonetische Bedeutung verloren, so daß Zeichen trotz gleicher phonetischer Einheit verschieden ausgesprochen werden. Genauso haben viele Zeichen eine gravierende Bedeutungswandlung erlebt, so daß die Bedeutungseinheit oft fehl am Platz ist. Der größte Teil, nämlich ca. 80 % der chinesischen Zeichen fällt in diese Entwicklungskategorie.

5. Kategorie notativer Ähnlichkeit:
   Ähnliche Begriffe werden mit Zeichen, die mit verwandten Bildungsvorschriften generiert wurden, gebildet. Daraus erhält man bei ähnlichen Bedeutungseinheiten ähnlich aussehende Zeichen. Beispielsweise haben 顶 Spitze und 颠 Gipfel eine verwandte Bedeutung und enthalten beide auf der rechten Seite das Radikal 页(ca. 1-2 %).

6. Kategorie geliehener Zeichen:
   Für neue Wörter in der gesprochenen Sprache, für die es keine schriftliche Repräsentation gab, wurden Schriftzeichen von anderen Wörtern mit ähnlicher oder gleicher Aussprache verwendet. Eines oder beide Zeichen wurden dann, um eine Unterscheidung zu ermöglichen, verändert.
   Beispiel: 其 qi2 ihr, sein und 箕 ji1 Kehrrichtschaufel, wobei Kehrrichtschaufel das ältere Zeichen ist und später von 其 nach 箕verändert wurde, um eine Unterscheidung zu ermöglichen. Dies ist das flexibelste Verfahren, aber auch dasjenige, das die ursprüngliche

Bedeutung der Zeichen am wenigsten erhält. Wiederum ca. 1-2 % der Zeichen fallen in diese Kategorie.

Die sich so entwickelten chinesischen Zeichen haben in der vergangenen Zeit allerlei Formänderungen durchlebt. Man unterscheidet hauptsächlich fünf verschiedene Phasen [2][4]:

1. Knochen- und Schildkrötenpanzer - Inschriften:
   Aus der späten Shang-Dynastie (1711-1066 v. Chr.) sind ca. 150000 Funde von Knocheninschriften bekannt und analysiert worden. 4500 verschiedene Zeichen sind in dieser Zeit verwendet worden, wovon von ca. 900 die Linie bis zu den heutigen Zeichen verfolgt werden kann. Die einzelnen Zeichen tauchten in vielen Varianten auf: Verschiedene Strichanzahl, Orientierung, Größe, so daß eine sichere Klassifikation nicht immer erfolgen kann.

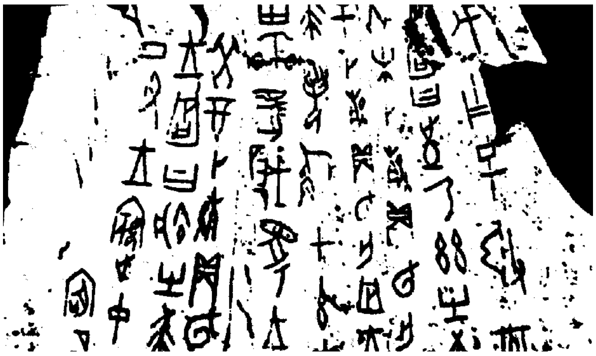


Abbildung 3.3: Orakelknocheninschrift

2. Bronzeinschriften:
   Während der Zhou-Dynastie (1066-256 v. Chr.) wurden die meisten Schriftfunde auf Bronzetafeln gefunden. Die Zeichenformen variierten in noch stärkerem Maße, während nach und nach immer weniger reine Piktogramme verwendet wurden.

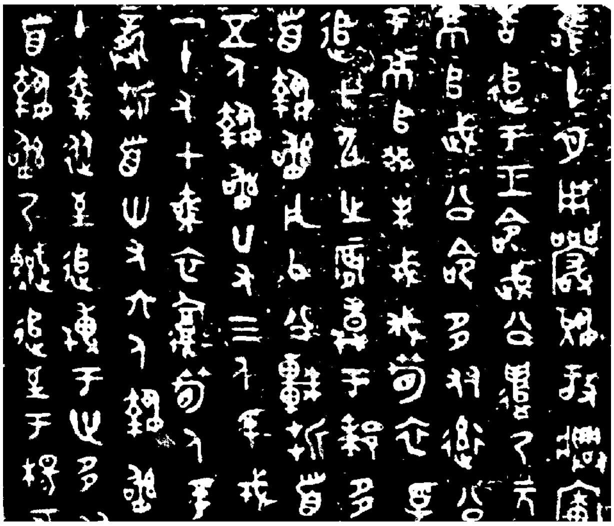


Abbildung 3.4: Bronzeinschrift der Zhou-Dynastie

3. Die Siegelschrift:
   Während der Qin-Dynastie (221-206 v. Chr.) entstand, unter Bemühung zur Vereinheitli-

chung, die Siegelschrift. Diese wurde dann zum Standard im gesamten Land erklärt. Bei den Zeichen dieser Schrift herrschten nun lineare, symbolische Formen vor, die noch viele runde Elemente enthielten. Es wurde die einheitliche, quadratische Form der Zeichen vorgegeben.

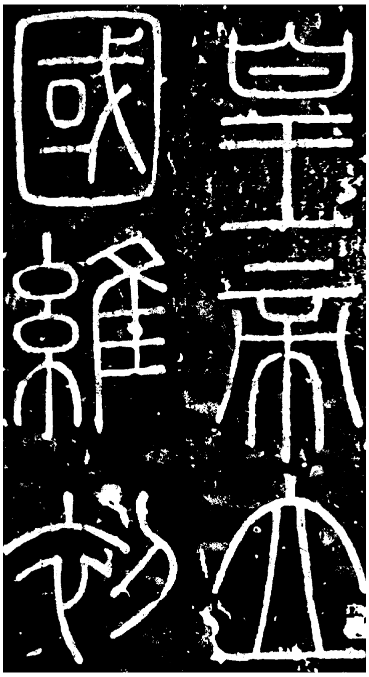


Abbildung 3.5: Siegelinschrift

4. Die offizielle Schrift:
   In der späten Qin- und der Han-Dynastie (206 v. Chr. bis 220 n. Chr.) wurde die ‚offizielle' Schrift eingeführt. Ziel war es ein schnelleres Schreiben zu ermöglichen, dazu wurden alle Kurven und Rundungen begradigt. Viele ähnliche Zeichenkomponenten wurden zusammengefaßt und, wo möglich, die Strichanzahl vermindert. Auch die Aussprache der einzelnen Zeichen wurde, wo man es für nötig hielt, geändert. Diese Schriftreform war die größte seit den frühen Anfängen der chinesischen Schrift. Sie war aber so konsequent, daß sich die Schrift in den nächsten 1800 Jahren bis heute nur noch geringfügig änderte.

說到底，出走的深層原因是學生長期的心理障礙得不到有效的疏導，中國的教育（無論是家庭或是學校）又歷來不重視學生心理素質的培養。一位心理學家指出，少男少女正處在心理成熟過{竣云滿岑_乳期”，其心理特點是渴望長大又\簡璊ㄓF幼稚心態的束縛，所以往往也是“危險期”。不管是來自哪一方面的誤導，都可能在他們的心路歷{竣手握U深深的烙印！

Abbildung 3.6: Offizielle Schrift wie sie heute noch in Taiwan verwendet wird

5. Standardisierte (vereinfachte) Form:
   Das Komitee zur Reform der Chinesischen Sprache, hatte in der Zeit von 1956-64 eine Reform beschlossen, in der folgende Vereinfachungen vorgenommen wurden:
   Verschiedene Zeichen mit derselben Bedeutung wurden meistens eliminiert, wobei das häufiger vorkommende Zeichen oder das einfachere Zeichen übrig geblieben ist und gege-

benenfalls noch weiter vereinfacht wurde. Dadurch wurde die Anzahl der häufig verwendeten Zeichen nahezu halbiert.

Diese Änderungen wurden aber nur auf dem Festland China durchgeführt. Hongkong, Taiwan und die meisten der Auslandschinesen benutzen die nicht vereinfachte Form, die auch als „Lang-Zeichen“-Form im Gegensatz zur „Kurz-Zeichen“-Form bezeichnet wird. Es existieren also heute zwei verwendete chinesische Schriftsysteme, wobei die Bedeutung der „Kurz-Zeichen“ wahrscheinlich mit der Bedeutung der Volksrepublik China weiter zunehmen wird. Für „Kurz-Zeichen“- Kundige ist es einfacher „Lang-Zeichen“- Texte zu lesen als umgekehrt. Von dieser Reform sind ca. 2500 Zeichen betroffen.

Die Regierung der Volksrepublik China hatte im Jahre 1974 erneut versucht die chinesische Schrift zu reformieren. Die Einführung dieser erneuten Reform stieß allerdings bei der Bevölkerung auf Ablehnung und mußte einige Jahre später wieder aufgehoben werden. (Manchmal findet man neben den standardisierten Zeichen noch Fragmente der erneuten Reform, die einem als Sprachschüler das Leben schwer machen können.)

说到底，出走的深层原因是学生长期的心理障碍得不到有效的疏导，中国的教育（ 无论是家庭或是学校 ）又历来不重视学生心理素质的培养。一位心理学家指出，少男少女正处在心理成熟过程中的“断乳期”，其心理特点是渴望长大又摆脱不了幼稚心态的束缚，所以往往也是“危险期”。不管是来自哪一方面的误导，都可能在他们的心路历程中打下深深的烙印！

Abbildung 3.7: Vereinfachte Schrift wie sie heute in der VR China verwendet wird.

(der dargestellte Text entspricht dem in der vorherigen Abbildung)

Systematik hatte im alten China einen großen Stellenwert, so daß es schon früh Sammlungen (Zeichen-Wörterbücher) der Zeichen gab. Dies wurde mitunter dadurch begünstigt, daß schon früh verschiedene Arten des Druckhandwerks eingeführt wurden. Um die Entwicklung der Anzahl der Zeichen aufzuzeigen, möchte ich nach [4] stellvertretend vier Wörterbücher erwähnen:

| **Name des (Zeichen-)Wörterbuchs** | **Veröffentlichungs-jahr** | **Anzahl der Zeichen** |
|---|---:|---:|
| Shou1Wen2Jie3Zi4 (说文接字) | ca. 100 | 9353 |
| Guang3 Yun4 (广韵) | ca. 1008 | 26000 |
| Kang1xi1 Zi4dian3 (康熙字典) | ca. 1716 | 47035 |
| Han4yu3 Da4 Zi4dian3 (汉语大字典) | 1997 | ca. 60000 |

## 3.2 Morphologie der chinesischen Sprache

In der frühen chinesischen Sprache repräsentierte ein Zeichen eine Sinneinheit, also genau ein Wort. Dies führte zu einer enormen Ansammlung von Zeichen, von denen einige so speziell waren, daß sie evtl. nur wenige Male in der Literatur aufgetaucht sind. Zum Beispiel gab es für den Begriff Pferd mehr als hundert Ausprägungen, von denen einige Zeichen sehr außergewöhnlich oder speziell waren:

- Pferd mit weißer Stirn
- Pferd mit schwarzen Lippen

- Pferd mit gelber Mähne
- Pferd mit einem weißen Auge
- Pferd, daß sieben Ellen groß ist
- Pferd mit blut-schwarzer Farbe
- ...

Durch diese Problematik, welche es äußerst schwer machte, daß außerhalb der Gelehrten oder Beamten die allgemeine Bevölkerung die Zeichenschrift lernte, wurde es erforderlich ein weiteres strukturelles Element einzuführen. Wie in einer Schrift, die auf einem Alphabet basiert, wurden Begriffe gebildet, indem mehrere Zeichen zusammengefaßt wurden. Dies erlaubte es nun einen beliebig großen Wortschatz bei begrenzter Zeichenanzahl zu bilden. Heutzutage sind ca. 90 % aller Wörter aus bis zu 10 morphemen Einheiten (Zeichen) zusammengesetzt, wobei die mittlere Wortlänge ca. 2 Zeichen beträgt.

Beispiele:
- fliegen 飞 + Maschine 机 = Flugzeug 飞机
- schlagen 打 + elektrisch 电 + Sprache 话 = telefonieren 打电话
- knüpfen 结 + Frucht 果 = 结果 Ergebnis

Ein weiteres Beispiel soll, anhand eines Satzes von Konfuzius, den Unterschied vom früheren zum heutigen, die Morphologie verwendenden Chinesisch, verdeutlichen:

| | ungetrennt | Sinneinheiten (Worte) getrennt |
|---|---|---|
| Konfuzius | 学而是习之，不亦说乎？ | 学 而 是 习 之， 不 亦 说 乎 ？ |
| heute | 有时候学习很有意思？ | 有时候 学习 很 有意思 ？ |

(Deutsch: Ist Lernen nicht hin und wieder ein Vergnügen?)

Wie man sieht war in früherer Zeit das Trennen von Bedeutungseinheiten (Wörter) identisch mit dem Trennen von Zeichen, was heute nicht mehr so gilt.

## 3.3 Das gesprochene Chinesisch

Die Aussprache des antiken Chinesisch hat sich bis zur Neuzeit stark gewandelt. Über die gesprochene chinesische Sprache in der frühen Zeit läßt sich nur sehr wenig sagen. Auch nach Aufkommen und Standardisierung der Schrift änderte sich die Lautgebung sehr stark. Somit sind Aussagen über die Lautgebung schwierig. Hilfreich für die Erforschung der frühen Phonetik ist die Poesie, in der besonders Reime und andere strukturelle Elemente Rückschlüsse auf die Aussprache zulassen. Diese Problematik ist unter anderem darauf zurückzuführen, daß die chinesischen Schriftzeichen sehr wenige Festlegungen über die Aussprache eines Wortes machen. Dies ermöglichte es sogar, daß völlig verschiedene, unabhängige Sprachen sich das chinesische Schriftsystem zu nutze machten. So kommt es zu dem Phänomen, daß sich zwei Sprecher völlig unterschiedlicher Sprachen über das Schriftsystem verständigen können. Japanisch und zum Teil Koreanisch sind Beispiele für solche Sprachen. Die althebräische Konsonantenschrift, in der die Vokale aus Effizienzgründen weggelassen wurden, ist ein ähnlicher Fall, in dem die Aussprachen von verschiedenen Worten nicht mehr ermittelt werden können.
Neben den Schwierigkeiten die phonetische Entwicklung in der Geschichte zu verfolgen, ergeben sich weitere, wenn man festlegen möchte, was denn die heute gesprochene chinesische Sprache ist. Aufgrund der Eigenschaften des Chinesischen ist es möglich, daß auch viele

„Dialekte“ dasselbe Schriftsystem benutzen können. Dabei haben viele Dialekte größere sprachliche Unterschiede als z.B. Deutsch und Englisch. Das bedeutet, daß Menschen eines Landes sich nicht sprachlich verständigen können, obwohl sie vielleicht dasselbe Schriftsystem beherrschen. Um ein Land einfach verwalten zu können und effektiv Radio- und Fernsehprogramme produzieren zu können, hat die Regierung der Volksrepublik China schon seit längerem damit begonnen, einen Dialekt, und zwar den in der Umgebung von Peking gesprochenen Mandarin-Dialekt, zur Hochsprache (Putonghua 普通话) zu deklarieren. In Schulen und allen öffentlichen Einrichtungen wird dieser Dialekt ausschließlich verwendet. Trotzdem gibt es noch viele Chinesen, die diese Hochsprache nicht verstehen und noch mehr, die sie nicht sprechen können. Die folgende Tabelle listet die wichtigsten chinesischen Dialekte nach Statistiken [13] auf, wobei die gesamte Anzahl der Sprecher sich bis heute allerdings noch gesteigert hat:

Sprecheranzahl in Millionen

| | 1953 | ca.1988 |
|---|---:|---:|
| Nordchinesische Dialekte (4 Gruppen) | 387 | 836 |
| Wu-Dialekte (4 Gruppen) | 46 | 77 |
| Dialekte von Guangdong | 27 | 47 |
| Dialekte von Hunan und Jiangxi | 26 | 56 |
| Hakka-Dialekte | 20 | 27 |
| Dialekte von Süd-Fujian | 15 | 25 |
| Dialekte von Nord-Fujian | 7 | 11 |
| **Gesamtzahl** | ***528*** | ***1079*** |

Zusätzlich gibt es noch Sprecher der chinesischen Sprache in Taiwan, Hongkong/Macao, in chinesischen Kolonien und Auslandschinesen in Nordamerika und Europa.

Neben diesen chinesischen Dialekten gibt es aber noch nichtchinesische Nationalitäten mit ihrer eigenen Sprache in der Volksrepublik China. Die folgende Tabelle gibt über diese Sprachfamilien eine Übersicht (offizielle Zahlen von 1957):

| **Name** | **Ethno-linguistische Gruppe** | **Siedlungsgebiet** | **Anzahl** |
|---|---|---|---:|
| Zhuang | Thai | Yunnan, Guangdong | 7800000 |
| Uiguren | Türkische Gruppe | Xinjiang, West-Gansu | 3900000 |
| Yi | Tibeto-birmanische Gruppe | Yunnan, Guizhou, Hunan | 3260000 |
| Zang | Tibetische Gruppe | Tibet, Qinghai, Sichuan | 2770000 |
| Miao | Miao-Yao | SW-Provinzen | 2680000 |
| Mandschuren | Tungusische Gruppe | NO, Mongolei, Pekinger Region | 2430000 |
| Mongolen | Mongolische Gruppe | Mongolei, NO, Gansu, Qinghai | 1640000 |
| Buyi | Thai | Yunnan | 1320000 |
| Koreaner | Koreanische Gruppe | Nordosten | 1250000 |

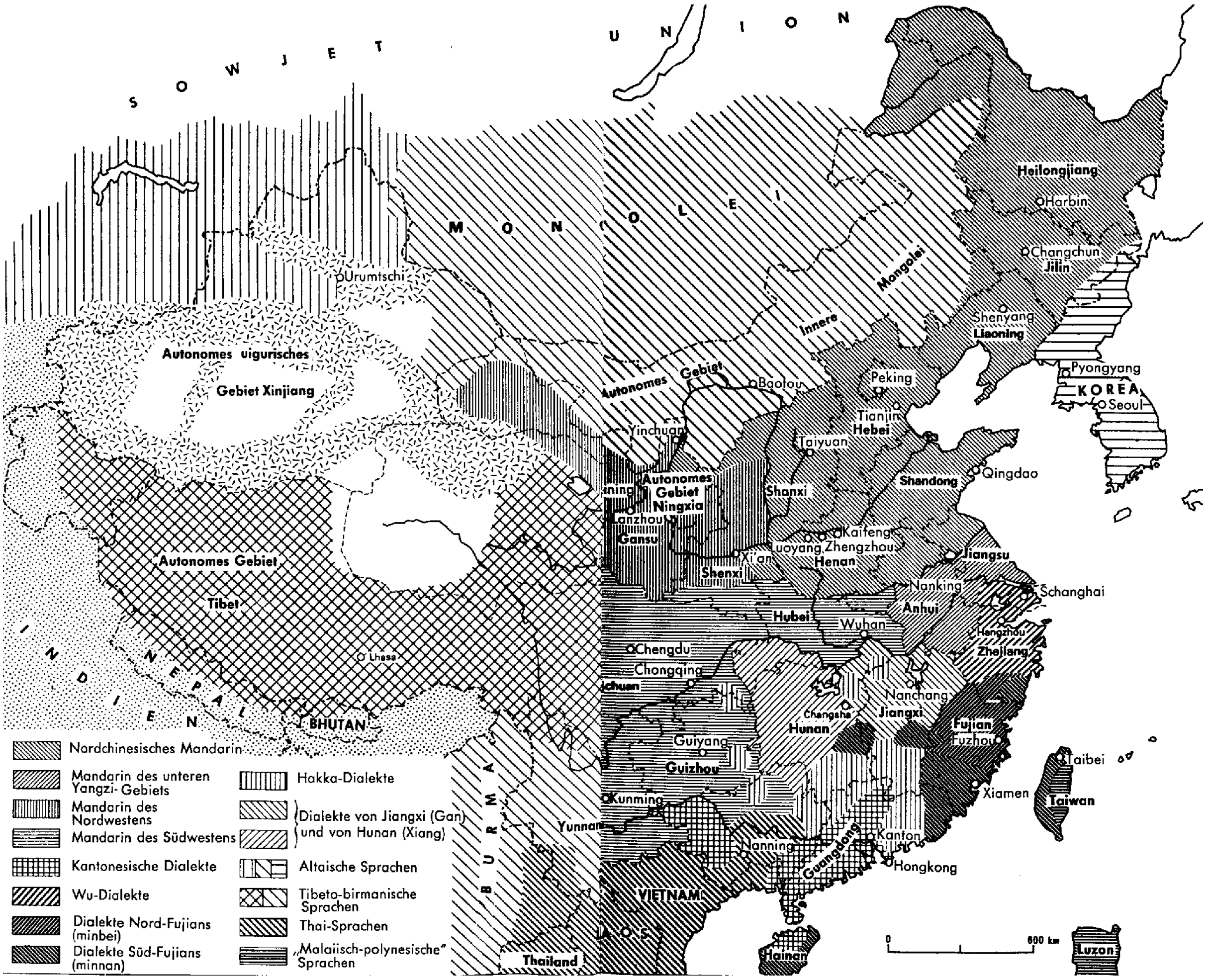

SOWJET UNION
MONGOLEI
Autonomes uigurisches Gebiet Xinjiang
Urumtschi
Autonomes Gebiet Tibet
Lhasa
NEPAL
BHUTAN
INDIEN
BURMA
Heilongjiang
Harbin
Changchun
Jilin
Shenyang
Liaoning
Pyongyang
KOREA
Seoul
Innere Mongolei
Autonomes Gebiet
Baotou
Peking
Tianjin
Hebei
Taiyuan
Shanxi
Shandong
Qingdao
Yinchuan
Autonomes Gebiet Ningxia
Lanzhou
Gansu
Kaifeng
Luoyang
Zhengzhou
Henan
Xi'an
Shenxi
Jiangsu
Nanking
Anhui
Schanghai
Hubei
Wuhan
Hangzhou
Zhejiang
Chengdu
Chongqing
Sichuan
Nanchang
Changsha
Hunan
Jiangxi
Fujian
Fuzhou
Taibei
Taiwan
Xiamen
Guiyang
Guizhou
Kunming
Yunnan
Nanning
Guangdong
Kanton
Hongkong
VIETNAM
LAOS
Thailand
Hainan
Luzon
0 600 km
Nordchinesisches Mandarin
Mandarin des unteren Yangzi-Gebiets
Mandarin des Nordwestens
Mandarin des Südwestens
Kantonesische Dialekte
Wu-Dialekte
Dialekte Nord-Fujians (minbei)
Dialekte Süd-Fujians (minnan)
Hakka-Dialekte
Dialekte von Jiangxi (Gan) und von Hunan (Xiang)
Altaische Sprachen
Tibeto-birmanische Sprachen
Thai-Sprachen
„Malaiisch-polynesische" Sprachen

Abbildung 3.8: Die Verteilung der Dialekte in China (aus [13])

In dem Bemühen die Aussprache festzuhalten und niederzuschreiben wurden verschiedene Lautsprachesysteme eingeführt. Das Bopomofo-System benutzt für jedes Phonem ein eigenes Zeichen.

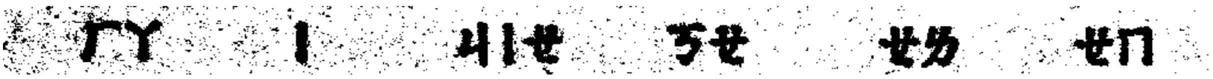

ㄏㄚ
ㄧ
ㄐㄧㄝ
ㄎㄝ
ㄝㄌ
ㄝㄇ

Abbildung 3.9: sechs Phoneme des zeichenbasierten Bopomofo-Systems

Weitere auf dem römischen Alphabet basierende Lautschriftsysteme wurden von Missionaren Anfang des neunzehnten Jahrhunderts, aus der Notwendigkeit die Sprache zu erlernen, eingeführt, wobei das Sprachempfinden jeweils stark von der eigenen Muttersprache geprägt war. Das Pinyin-Lautsprachesystem wurde von der Regierung der Volksrepublik China zur Vereinheitlichung der Aussprache entwickelt und als Standard zum Erlernen der Hochsprache (Putonghua 普通话) festgelegt. Die folgende Tabelle vergleicht drei auf Buchstaben basierende Systeme anhand von fünf Beispielen:

| Pinyin | Wade-Giles | Yale |
|---|---|---|
| nian4 | nien4 | nyan4 |
| piao4 | p'iao4 | pyau4 |
| xuan1 | hsu:an1 | sywan1 |
| zi5 | tzu5 | dz5 |
| zong1 | tsung1 | dzung1 |

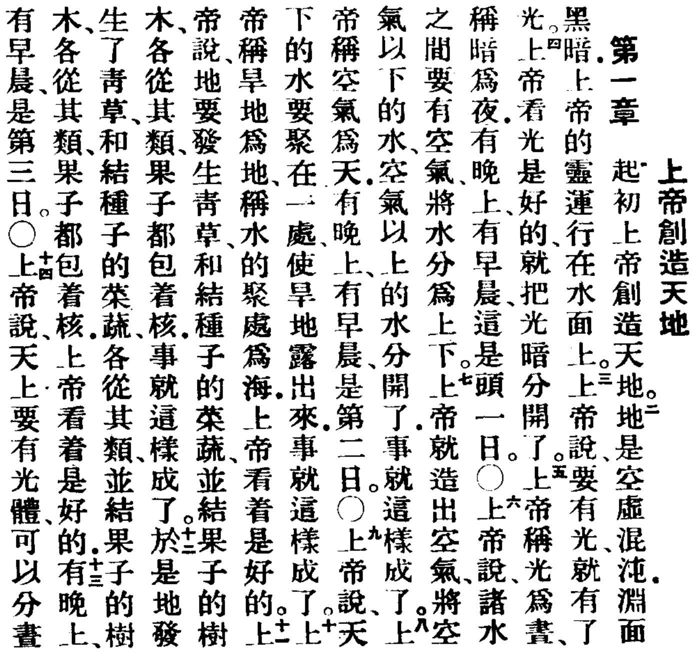
In Taiwan werden die Systeme Wade-Giles, Yale und Bopomofo nebeneinander benutzt, während die Volksrepublik das selbst entwickelte Pinyin-System nun mehr ausschließlich benutzt. Für die chinesische Spracherkennung wurde das Pinyin-System gewählt, wobei eine Überführung der einzelnen Systeme ineinander ohne größere Schwierigkeiten mit einfachen Tabellen erfolgen kann.

## 3.4 Weitere Besonderheiten der chinesischen Sprache

Die chinesische Sprache basiert auf einer großen Anzahl von Zeichen. Diese sind die kleinsten Sinneinheiten und die Grundlage zum Aufbau von Wörtern und Sätzen. Diese Zeichen füllen jeweils genau ein Quadrat und haben alle dieselbe Größe.

### 3.4.1 Die Schreibrichtung

Da jedes Zeichen für sich eine unveränderliche, abgeschlossene Einheit darstellt, ist der Anknüpfungspunkt für das nächste zu schreibende Zeichen beliebig. Es gab in der Vergangenheit deshalb auch die unterschiedlichsten Schreibrichtungen nebeneinander. In Zeitungen wurden sogar auf derselben Seite unterschiedliche Schreibrichtungen eingesetzt. Die Volksrepublik China verwendet nahezu nur noch die uns geläufige Schreibrichtung (links-nach-rechts, oben-nach-unten, vorne-nach-hinten). In Taiwan sind noch andere Schreibrichtungen gebräuchlich, wobei, durch Anforderungen der besseren Computerverarbeitbarkeit, immer mehr die uns geläufige Schreibrichtung verbreitet wird.



**上帝創造天地**

**第一章**

起初上帝創造天地。[二]地是空虛混沌．淵面黑暗．上帝的靈運行在水面上。[三]上帝說、要有光、就有了光。[四]上帝看光是好的、就把光暗分開了。[五]上帝稱光爲晝、稱暗爲夜．有晚上、有早晨、這是頭一日。〇[六]上帝說、諸水之間要有空氣、將水分爲上下。[七]上帝就造出空氣、將空氣以下的水、空氣以上的水分開了．事就這樣成了。[八]上帝稱空氣爲天．有晚上、有早晨、是第二日。〇[九]上帝說、天下的水要聚在一處、使旱地露出來．事就這樣成了。[十]上帝稱旱地爲地、稱水的聚處爲海．上帝看着是好的。[十一]上帝說、地要發生青草、和結種子的菜蔬、並結果子的樹木、各從其類、果子都包着核．事就這樣成了。[十二]於是地發生了青草、和結種子的菜蔬、各從其類、並結果子的樹木、各從其類、果子都包着核．上帝看着是好的．[十三]有晚上、有早晨、是第三日。〇[十四]上帝說、天上要有光體、可以分晝

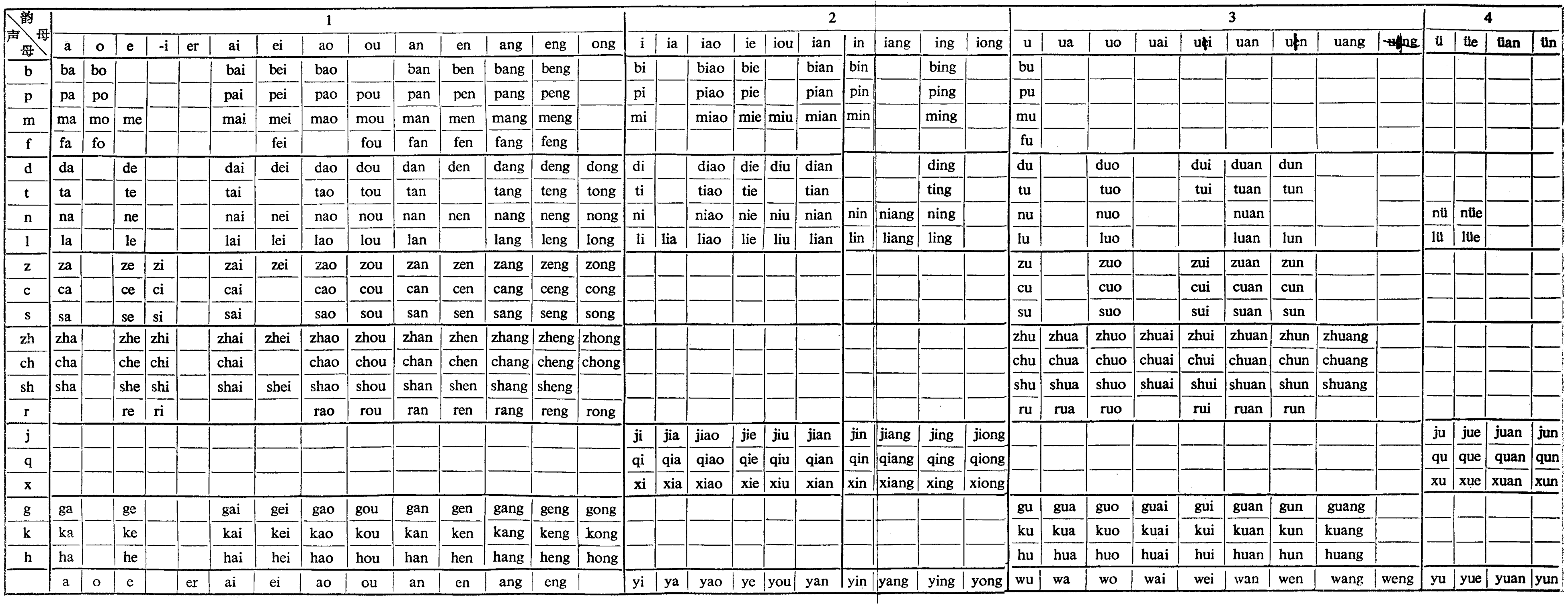
Abbildung 3.10: Beispiel einer in Taiwan gebräuchlich Schreibrichtungen

(1. Mose 1 einer taiwanesischen Bibel)
(oben-nach-unten, rechts-nach-links, im Buch von hinten-nach-vorne)

## 3.4.2 Die Phonetik und Tonalität

Die Wörter der chinesische Hochsprache können in Aussprachesilben unterteilt werden. Dabei ist es bemerkenswert, daß, von wenigen Ausnahmen einmal abgesehen, jedes Zeichen genau eine Silbe repräsentiert, wobei allerdings ein Zeichen je nach Kontext unterschiedliche Aussprachesilben repräsentieren kann.
Die Aussprachesilben bestehen aus einem Konsonanten (oder Affrikat = Verschlußlaut mit anschließendem Reibelaut) und einem Vokalkonstrukt, wenige auch nur aus einem Vokalkonstrukt.
Menge der Konsonanten = {b, p, m, f, d, t, n, l, z, c, s, zh, ch, sh, r ,j, q, x, g, k, h, y, w}
Menge der Vokalkonstrukte = {a, o, e, -i, er, ai, ei, ao, ou, an, en, ang, eng, ong,
i, ia, iao, ie, iou, ian, in, iang, ing, iong,
u, ua, uo, uai, ui, uan, un, uang, ü, üe, üan, ün}

Die Tabelle auf der folgenden Seite zeigt die erlaubten Kombinationen von Konsonanten und Vokalkonstrukten.



| 韵母<br>声母 | 1 | | | | | | | | | | | | | | 2 | | | | | | | | | | 3 | | | | | | | | | 4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | o | e | -i | er | ai | ei | ao | ou | an | en | ang | eng | ong | i | ia | iao | ie | iou | ian | in | iang | ing | iong | u | ua | uo | uai | u~~e~~i | uan | u~~e~~n | uang | ~~ueng~~ | ü | üe | üan | ün |
| b | ba | bo | | | | bai | bei | bao | | ban | ben | bang | beng | | bi | | biao | bie | | bian | bin | | bing | | bu | | | | | | | | | | | | |
| p | pa | po | | | | pai | pei | pao | pou | pan | pen | pang | peng | | pi | | piao | pie | | pian | pin | | ping | | pu | | | | | | | | | | | | |
| m | ma | mo | me | | | mai | mei | mao | mou | man | men | mang | meng | | mi | | miao | mie | miu | mian | min | | ming | | mu | | | | | | | | | | | | |
| f | fa | fo | | | | | fei | | fou | fan | fen | fang | feng | | | | | | | | | | | | fu | | | | | | | | | | | | |
| d | da | | de | | | dai | dei | dao | dou | dan | den | dang | deng | dong | di | | diao | die | diu | dian | | | ding | | du | | duo | | dui | duan | dun | | | | | | |
| t | ta | | te | | | tai | | tao | tou | tan | | tang | teng | tong | ti | | tiao | tie | | tian | | | ting | | tu | | tuo | | tui | tuan | tun | | | | | | |
| n | na | | ne | | | nai | nei | nao | nou | nan | nen | nang | neng | nong | ni | | niao | nie | niu | nian | nin | niang | ning | | nu | | nuo | | | nuan | | | | nü | nüe | | |
| l | la | | le | | | lai | lei | lao | lou | lan | | lang | leng | long | li | lia | liao | lie | liu | lian | lin | liang | ling | | lu | | luo | | | luan | lun | | | lü | lüe | | |
| z | za | | ze | zi | | zai | zei | zao | zou | zan | zen | zang | zeng | zong | | | | | | | | | | | zu | | zuo | | zui | zuan | zun | | | | | | |
| c | ca | | ce | ci | | cai | | cao | cou | can | cen | cang | ceng | cong | | | | | | | | | | | cu | | cuo | | cui | cuan | cun | | | | | | |
| s | sa | | se | si | | sai | | sao | sou | san | sen | sang | seng | song | | | | | | | | | | | su | | suo | | sui | suan | sun | | | | | | |
| zh | zha | | zhe | zhi | | zhai | zhei | zhao | zhou | zhan | zhen | zhang | zheng | zhong | | | | | | | | | | | zhu | zhua | zhuo | zhuai | zhui | zhuan | zhun | zhuang | | | | | |
| ch | cha | | che | chi | | chai | | chao | chou | chan | chen | chang | cheng | chong | | | | | | | | | | | chu | chua | chuo | chuai | chui | chuan | chun | chuang | | | | | |
| sh | sha | | she | shi | | shai | shei | shao | shou | shan | shen | shang | sheng | | | | | | | | | | | | shu | shua | shuo | shuai | shui | shuan | shun | shuang | | | | | |
| r | | | re | ri | | | | rao | rou | ran | ren | rang | reng | rong | | | | | | | | | | | ru | rua | ruo | | rui | ruan | run | | | | | | |
| j | | | | | | | | | | | | | | | ji | jia | jiao | jie | jiu | jian | jin | jiang | jing | jiong | | | | | | | | | | ju | jue | juan | jun |
| q | | | | | | | | | | | | | | | qi | qia | qiao | qie | qiu | qian | qin | qiang | qing | qiong | | | | | | | | | | qu | que | quan | qun |
| x | | | | | | | | | | | | | | | xi | xia | xiao | xie | xiu | xian | xin | xiang | xing | xiong | | | | | | | | | | xu | xue | xuan | xun |
| g | ga | | ge | | | gai | gei | gao | gou | gan | gen | gang | geng | gong | | | | | | | | | | | gu | gua | guo | guai | gui | guan | gun | guang | | | | | |
| k | ka | | ke | | | kai | kei | kao | kou | kan | ken | kang | keng | kong | | | | | | | | | | | ku | kua | kuo | kuai | kui | kuan | kun | kuang | | | | | |
| h | ha | | he | | | hai | hei | hao | hou | han | hen | hang | heng | hong | | | | | | | | | | | hu | hua | huo | huai | hui | huan | hun | huang | | | | | |
| | a | o | e | | er | ai | ei | ao | ou | an | en | ang | eng | | yi | ya | yao | ye | you | yan | yin | yang | ying | yong | wu | wa | wo | wai | wei | wan | wen | wang | weng | yu | yue | yuan | yun |

Abbildung 3.11: Tabelle der Pinyin - Sprachlaute des Mandarin Dialekts

Die Anzahl der verfügbaren Aussprachesilben, wie in Abbildung 2.11 ersichtlich, beträgt nur ca. 400. Diese Zahl ist im Gegensatz zur Gesamtanzahl der Zeichen von ca. 60000 sehr niedrig, so daß im Mittel 150 Zeichen dieselbe Aussprache haben. Um diese große Anzahl gleich ausgesprochener Zeichen doch ein wenig besser zu differenzieren, enthält jede Aussprachesilbe eine bestimmten Tonverlauf, d.h. eine Veränderung der Tonhöhe während der Aussprache. In der Hochsprache ist dies einer von fünf möglichen Tonverläufen, wobei der fünfte Tonverlauf einer Nichtbetonung entspricht. Andere chinesische Dialekte enthalten bis zu 13 Tonverläufe. Der Tonverlauf ist bedeutungstragend, somit ist bei gleicher Aussprachesilbe und anderem Tonverlauf ein völlig anderes Zeichen mit unterschiedlicher Bedeutung gemeint.

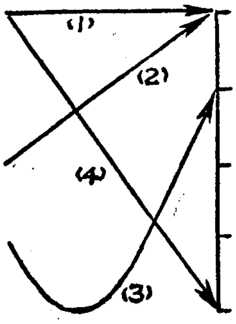


Abbildung 3.12: Die Tonverläufe der chinesischen Hochsprache

Die Pinyin-Lautschrift gibt beides wieder, die Aussprachesilbe und den Tonverlauf. Die in der Literatur verwendete Kennzeichnung des Tonverlaufs (oder kurz Ton genannt) könnte auf Computern nur unter zu Hilfenahme von speziellen Zeichensätzen erfolgen und wird deshalb der Einfachheit halber durch Zahlen gekennzeichnet (dies erleichtert auch die Computereingabe).

Beispiel: die 5 Töne einer Aussprachesilbe mit einer Auswahl von Bedeutungen:

- mā = ma1 : Mutter 妈, streicheln 摩, Dämmerung 麻, wischen 抹, ...
- má = ma2 : Hanf 麻, Anästhesie 麻, Kröte 蟆, ...
- mǎ = ma3 : Pferd 马, Nummer 码, stapeln 码, Yard 码, Ameise 蚂, ...
- mà = ma4 : schimpfen 骂, Heuschrecke 蚂, ...
- ma = ma5 : Fragepartikel 吗, Bewußtseinskennzeichnung 嘛, ...

Trotz der Tonalität der Sprache ergeben sich für eine Aussprachesilbe mit Tonverlauf oftmals noch Dutzende von möglichen Zeichen und damit auch Bedeutungen. Das heißt, daß beim Vorlesen, also der Transformation von Schrift in gesprochene Sprache, Information verloren geht, die nur aus dem Satzkontext und Metawissen wiedergewonnen werden kann. Chinesische Zeichen und auch Wörter sind also sehr stark vom Kontext abhängig, was im Deutschen nur sehr viel seltener der Fall ist.

Beispiel: Auch im Deutschen kann Kontextwissen erforderlich sein:
Er geht zur Bank.
➔ Kann doppeldeutig sein: Geht er zum Geldinstitut oder zur Sitzbank?
Er sitzt auf der Bank, um sich auszuruhen | Er geht zur Bank, um ein Konto zu eröffnen.
➔ Bedeutung kann mit großer Wahrscheinlichkeit aus dem Kontext erschlossen werden.

Die Tonalität selbst ist wiederum geringfügig vom Kontext abhängig. Aber die Regeln zur Änderung von Tönen je nach Kontext weichen bei unterschiedlichen Lehrbüchern wiederum voneinander ab. Sie werden in dieser Arbeit nicht berücksichtigt.

### 3.4.3 Das chinesisches Zahlensystem

Das chinesisches Zahlensystem unterscheidet sich in zweierlei Hinsicht von unserem. Zum einen werden statt der arabischen Ziffern 0 1 2 3 4 5 6 7 8 9 ... die chinesischen Zeichen 零（〇）一二三四五六七八九十百千万 usw. verwendet und zum anderen basiert das Zusammenfassen großer Zahlen nicht auf der Basis 1000 sondern auf der Basis 10000.

Beispiel: Hundertmillionen (100 * 1000 * 1000) entspricht 千万 (1000 * 10000)

Dies führt manchmal beim Kontakt mit Chinesen zu Verwechslungen. Heute werden aber auch immer mehr arabischen Ziffern verwendet. Besonders in der Wissenschaft, als Artikelnummern und zur mathematischen Berechnung. Weiterhin gibt es noch ein älteres, selten noch gebrauchtes Zahlensystem, welches in einigen Fällen alternativ verwendet wird.

### 3.4.4 Die Grammatik der chinesischen Sprache

Die chinesische Grammatik benutzt keine so ausdrucksstarken Methoden wie Konjunktion und Deklination, sondern muß viele grammatische Strukturen über Funktionalzeichen und Konstrukte realisieren. Dabei spielen Funktionswörter, die manchmal eine Art Klammerstruktur haben, eine wesentliche Rolle. Sie realisieren die Funktionalität, die in westlichen Sprachen der Grammatik zukommt. Im modernen Chinesisch sind das 971 oft gebrauchte Funktionswörter, die sich in 572 Wörter mit adverbialer Funktion, 147 Konjunktionen, 47 Partikel, 43 Interjektionen, 67 Strukturen und 95 mit sonstigen Aufgaben unterteilen [16].
Eine Behandlung der unerschöpflichen chinesischen Grammatik würde mir Sicherheit den Rahmen dieser Arbeit sprengen. Eine gute Einführung ist in [9] und [42] zu finden.

### 3.4.5 Sprichwörter (成语)

In der gehobenen chinesischen Ausdrucksweise kommt der Verwendung von Sprichwörtern eine besondere Bedeutung zu. Diese Sprichwörter (成语 cheng2yu3) haben ihren Ursprung meist in der klassischen chinesischen Literatur und stammen dort von einer speziellen Geschichte oder einer Anekdote. Sie bestehen meist aus genau vier Zeichen, die, ohne einen Satz zu bilden, abstrakt eine Aussage über einen Sachverhalt wiedergeben.
Um gehobenes Chinesisch verstehen oder sprechen zu können, ist es nötig eine größere Auswahl der mehreren tausend Sprichwörter zu kennen.

Beispiel einer Anekdote mit abgeleitetem Sprichwort:

=> Sprichwort: 杯弓蛇影 (Glas Bogen Schlange Schatten)

晋朝的时候，有一个人叫乐广，他很会说话， 很能用道理说服人。

乐广有一个好朋友，两个人常常在一起喝酒，谈天。可是后来，那个朋友有一个多月没到乐广家来了。乐广就派人去了解情况。派去的人回来说，那个朋友病了。原来上一次他在乐广家喝酒，看见酒杯里有一条小蛇，可是酒已经喝下去了，有什么办法呢！他当时心里很不舒服， 回到家就病了。
乐广听了，觉得很奇怪，酒杯里怎么会有小蛇呢？他走到上次喝酒的地方，仔仔细细地看了一遍，忽然看见墙上挂着一张弓，他立刻明白了。于是，他又派人去请那个朋友来喝酒，而且还说他能治好他的病。
那个朋友开始恨不愿意来，最后他还是来了。乐广还请他喝酒，让他坐老地方。那个朋友本来就很不妨心，他往酒杯里一看，嘿，那条小蛇还在酒杯里呢！他吓得出了一身冷汗。乐广指着墙上的弓笑着说：“酒杯里没有什么蛇，这是墙上弓的影子。”
他把墙上的弓拿下来，酒杯里的小蛇立刻不见了。他朋友这才明白是怎么一回事，病也就好了。

Übersetzung:

In der Jin-Dynastie gab es einen Mann namens Yueguang, der sehr redegewandt war und mit Argumenten überzeugen konnte.
Yueguang hatte einen guten Freund mit dem er oft zusammensaß und beim Schnapstrinken über Gott und die Welt redete. Aber eines Tages blieb jener Freund mehr als einen Monat aus. Yueguang war besorgt und schickte jemanden um in Erfahrung zu bringen, was denn los sei. Als der Bote zurückkam, berichtete er, daß jener Freund seitdem er das letzte Mal bei Yueguang Schnaps trinken war, krank ist, weil in dem Schnapsglas eine kleine Schlange war. Als er sie entdeckte war es schon zu spät, denn er hatte schon davon getrunken - er konnte also nichts mehr machen. Von dieser Zeit ging es ihm dann immer schlechter.
Als Yueguang das hörte, war ihm seltsam zumute. Wie konnte in den Schnaps eine kleine Schlange geraten? Er ging an den Ort an dem sie das letzte Mal zusammen getrunken hatten zurück und untersuchte alles einmal sorgfältig. Plötzlich sah er an der Wand einen Bogen hängen und sofort wurde ihm einiges klar. Wieder schickte er jemanden zu seinem Freund, daß er wieder zum Schnapstrinken kommen soll - und daß, wenn er kommt, er ihn auch von seiner Krankheit heilen könne.
Jener Freund wollte anfangs überhaupt nicht kommen. Am Ende kam er trotzdem, weil er aus Verzweiflung nicht mehr wußte was er tun soll. Yueguang bot ihm wieder Schnaps am selben Ort wie letztes Mal an. Der Freund fühlte sich immer unwohler in seiner Haut. Als er dann ins Schnapsglas schaute sah er wieder die kleine Schlange und erschrakt fürchterlich, daß ihm der kalte Schweiß aus allen Poren kam.
Yueguang zeigte zum Bogen an der Wand und beruhigte ihn: „Die Schlange in deinem Glas ist nur der Schatten des Bogens an der Wand.“ Er nahm den Bogen von der Wand und sofort verschwand die Schlange im Glas. Jetzt begriff der Freund endlich die gesamte Sache und seine Krankheit war auch geheilt.

Bedeutung und Anwendung:
Dieses Sprichwort wird für Menschen benutzt, die alles überängstlich ohne Grund anzweifeln.
(这个成语用来比喻有人怀疑这个，怀疑那个，实际并没有那么一回事。)

Die Bedeutung und Schwierigkeit chinesischer Sprichwörter geht weit über die Bedeutung der deutschen Sprichwörter hinaus, da sie

a) häufiger benutzt werden (im gehobenen Chinesisch)
b) selten auf den Sinn geschlossen werden kann (im Deutschen ist der Bezug zur Bedeutung meist direkter z.B. Lügen haben kurze Beine, Morgenstund hat Gold im Mund, ..... )
c) aus klassischem Chinesisch abgeleitet sind
d) keine grammatische Struktur besitzen
e) meist nur aus Wortbruchteilen bestehen

### 3.4.6 Sonstiges

- Städtenamen: Manche Städtenamen sind sehr alt und benutzen sehr seltene Zeichen, die oft vielen Chinesen unbekannt oder in manchen Computerzeichensätzen nicht vorrätig sind.
- Personennamen: Der Nachname kommt vor dem Vornamen bei der Anrede. Üblich ist es immer sowohl Nachname als auch Vorname zu nennen. Das gilt sogar für Freunde und teilweise auch Familienangehörige. Namenszeichen können auch sehr selten sein.
- Chinesischer Kalender: Der chinesische Kalender basiert auf den Mondphasen. Dadurch ist der Jahresbeginn um eine sich ändernde Anzahl von Tagen gegenüber unserem Jahresbeginn verschoben. Neben dem chinesische Kalender wird auch der westliche Kalender verwendet.

## 3.5 Chinesisch und Computer

Seit Einführung der Drucktechnik, die viel früher als in der westlichen Welt eingeführt wurde, war man bemüht diese zu verbessern. Dies führte im letzten Jahrhundert dann zur Entwicklung von chinesischen Schreibmaschinen, die sich aufgrund der Komplexität, Größe und aufwendigen Bedienung nie durchsetzten konnten. Es konnte immer nur ein Teil der Zeichen direkt gedruckt werden (max. 2000). Alle anderen, selteneren Zeichen mußten von Hand aus einem Kasten herausgesucht und auf eine Vorrichtung aufgesteckt werden. In den letzten Jahrzehnten, mit Einführung des Computers, änderte sich diese Situation dann schlagartig. Der kostengünstigere Computer ermöglichte es, alle Zeichen gleichartig zu behandeln und mittels Matrixdruckern auszugeben. Trotz dieser Vorteile ist die Verarbeitung der chinesischen Schrift mittels den, hauptsächlich im Westen entwickelten Computern, nicht so ohne weiteres möglich. Es stellen sich die in den folgenden Unterkapiteln behandelten Fragen nach der Eingabemethode und der Codierung chinesischer Zeichen.

### 3.5.1 Eingabemethoden

In China werden Computer mit einer Standardtastatur ausgeliefert, denn eine spezielle Tastatur mit allen chinesischen Zeichen wäre zu teuer, zu unübersichtlich und zu groß.
Es müssen deshalb Verfahren entwickelt werden, um die chinesischen Zeichen auf die wenigen Tasten der Tastatur abzubilden. Neben Verfahren, die die Tastatur als Eingabeinstrument benutzen, existieren auch andere Hilfsmittel. Insgesamt haben sich eine beträchtliche Anzahl von Eingabemethoden für die unterschiedlichsten Einsatzgebiete entwickelt.

Beispiele:

- Pinyin-Eingabemethode: Bei dieser Eingabemethode gibt man über die Tastatur die Aussprache eines Zeichens in der Pinyin-Lautschrift plus Ton ein. Der Computer generiert dann eine Liste von in Frage kommenden Zeichen, aus denen der Benutzer dann das richtige auswählen kann. Wenn man die Pinyin-Lautschrift beherrscht, was bei älteren Chinesen nur selten der Fall ist, kann man diese Eingabemethoden sehr schnell erlernen. Statistische oder neuronale Methoden helfen bei manchen Systemen die Auswahlliste möglichst optimal zu sortieren, so daß oft das erste vorgeschlagene Zeichen benutzt werden kann und der Suchaufwand sich stark reduziert. Trotzdem ist die Eingabegeschwindigkeit von Texten relativ niedrig. Bei Zeichen, deren Aussprache nicht genau bekannt ist, muß zusätzlich noch in einem Aussprachewörterbuch nachgeschlagen werden.

- Zeichenzerlegende Eingabemethoden: Von diesem Typ gibt es sehr viele unterschiedliche Eingabemethoden. Meist wird aus der Struktur der Zeichen ein Zahlen- oder Zeichencode erzeugt, der das Zeichen repräsentiert. Diese Form der Eingabe erfordert einen großen Lernaufwand von Codes, die je nach der verwendeten Methode teils eine sehr schnelle Eingabe von geübten Personen erlauben.
- Pen-Eingabe (Online OCR): Eine größere Bedeutung hat die Eingabe mittels eines Pens erlangt. Bei dieser Methode werden Zeichen auf ein spezielles Tableau geschrieben, welche mittels Software ausgewertet werden. Dieses Verfahren erfordert allerdings zusätzliche Hardware und Übung im gleichmäßigen und computerleserlichen Schreiben. Nach einer Trainingsphase ist die Erkennungsrate für Gelegenheitsschreiber im allgemeinen aber sehr hoch (ca. 95 –99%). Neben der Bitmap des eigentlichen Zeichens werden auch Strichrichtung, Strichreihenfolge und Strichanzahl ausgewertet. Somit ist es wesentlich einfacher möglich, ein einzelnes von Hand geschriebenes Zeichen zu erkennen.
- OCR-Eingabe: Mittels eines Scanners werden Texte als Bilder zum Computer übertragen, wo sie dann in Zeichen umgesetzt, d.h. in einzelne Teile partitioniert und diesen dann Zeichen zu geordnet werden. Dieses Verfahren eignet sich nur für gedruckte Texte, da OCR-Verfahren für chinesische Handschrift noch nicht weit genug entwickelt sind. Die chinesische handgeschriebene Schnellschrift weicht sehr stark von der Druckschrift ab, da der Schreiber weitreichende Variationsmöglickeiten hat.
- Spracherkennung: Die Spracherkennung ist sicher die natürlichste Eingabemethode und würde es erlauben die Benutzerschnittstelle für alle Sprachen einheitlich zu gestalten. Aufgrund der Sonderrolle, der der automatischen Spracherkennung bei zeichenbasierten Sprachen wie auch Chinesisch als Eingabemethode zukommt, ist eine schnellere und breitere Verbreitung als in der westlichen Welt zu erwarten.
  Die weiteren Kapitel beschäftigen sich mit dem Aufbau eines solchen Systems.

### 3.5.2 Zeichendarstellung und Fonts

Die Zeichendarstellung auf dem Bildschirm und anderen Ausgabegeräten wie Druckern, erfordern besondere Maßnahmen:

- spezielle Grafiktreiber die mehr als 128/256 Zeichen darstellen können
- hochauflösende Bildschirme und Drucker
- Fontgrößen von evtl. mehreren Megabyte müssen verarbeitet werden können (TTF 1-4 MB)
- ein Zwei-Bytecode ist erforderlich um alle Zeichen codieren zu können
- ein Zeilenumbruch darf nur immer nach einer geraden Anzahl von Bytes geschehen
- grafikfähige Drucker mit genügend Hauptspeicher (mehr als 1 MB ist bei Laserdruckern erforderlich)

### 3.5.3 Codierung

Für die chinesische Schriftsprache existieren verschiedene Zwei-Bytecodierungen. Sie unterscheiden sich nach Anwendungsgebiet und Menge der darstellbaren Zeichen.

- Telegraphiecode: 1881 wurde in China der Telegraphiecode festgelegt, bei dem ein Codewort aus 4 Ziffern von 0000 bis 9999 besteht. Dieser Telegraphiecode enthält 10000 Zeichen und wird bis heute noch verwendet.
- GuoBiao: Offizielle Codierung der Volksrepublik China. Im Jahr 1981 eingeführt, enthält die Kodierung 7783 chinesische Zeichen. Dieser Standardschriftzeichencode (Guo2-Biao1) existiert in verschiedenen Varianten, da er mehrmals ergänzt und korrigiert wurde, was aber in der Praxis nur wenige Zeichen betrifft. Außer den chinesischen Zeichen enthält er zusätzlich die Kodierung der japanischen Lautschrift, Hiragana und Katakana, der kyrillischen Schrift, sowie der lateinischen Schrift. Insbesondere ist der GuoBiao Code eine Obermenge des Standard ASCII-Codes. Alle nicht ASCII-Zeichen entsprechen einem Wert größer 127, wobei sich dann 16384 nicht ASCII 2-Byte Zeichen darstellen lassen. Somit ist es möglich problemlos ASCII Zeichen mit chinesischen Zeichen in einem Text darzustellen.
  Wenn zwei aufeinander folgende Zeichen einen ASCII-Wert größer als 127 haben, werden sie als ein chinesisches Zeichen interpretiert, sonst als normales Zeichen. Dies führt in der deutschen Sprache dazu, daß zwar einzelne Umlaute und “ß” dargestellt werden können (ä, ü, ö, ß) aber nicht zwei aufeinander folgende Sonderzeichen.
- Big5: Der Big5-Code wird in Taiwan und Hongkong benutzt.
- Unicode: Die Unicode-Codierung wurde von Microsoft entwickelt, um die Lokalisierung von Windows zu vereinfachen [21]. Da Microsoft die Fortentwicklung von Unicode einem unabhängigen Gremium übergeben hat, ist eine plattformübergreifende Verbreitung sicher. Unicode wird bis jetzt zwar noch nicht oft angewendet, aber aufgrund seiner Flexibilität und Einfachheit wird er die länderspezifischen Kodierungen mittelfristig sicher ablösen. Auch die strikte Verwendung von Unicode in Java, Windows NT und Office 97 werden die Unicodeverbreitung schnell voranbringen.
  Der Unicode codiert mit einem Zwei-Byte Code die gebräuchlichsten Alphabete der Welt, wie das lateinische, griechische, kyrillische, hebräische, arabische, thailändische, mongolische, und viele andere, unter anderem auch einige exotische Alphabete, wie z.B. das Tibetische. Die asiatischen Schriftzeichencodes werden in der Unicode Terminologie als CJK-Gruppe (für China, Japan, Korea) bezeichnet. Der Unicode ermöglicht es in einem Dokument beliebige Schriften gemeinsam zu nutzen, sogar die Schreibrichtung innerhalb eines Dokument zu wechseln (z.B. von oben nach unten) was mittels Steuerkommandos realisiert wird. Diese Arbeit basiert auf Unicode, somit existieren keine Probleme mit Selbstlauten und der Mischung von Lang- und Kurzzeichen.
- hz: Die hz-Kodierung ist eine Codierung, in der die chinesischen Zeichen auf ASCII-Zeichen abgebildet werden, um problemlos Emails verschicken zu können. ‘~{‘ entspricht dabei der Kennzeichnung, daß alle folgenden Zeichen bis ‘~}~’ als chinesische Zeichen aufgefaßt werden sollen.

Um die einzelnen Codierungen ineinander zu überführen, gibt es im Public Domain- sowie im kommerziellen Bereich entsprechende Konverter unterschiedlicher Qualität.

### 3.5.4 Vorteile der chinesischen Schrift

Neben den Eigenschaften, die im Vergleich zu unserem Schriftsystem die Verwendung der chinesischen Schrift erschweren, gibt es auch einige Vereinfachungen und Vorteile:

- Keine Silbentrennung: Da es im Chinesischen weder Leerzeichen zwischen zwei Zeichen, noch Wortgrenzen gibt, kann nach jedem Zeichen die Zeile umgebrochen werden. Es müssen also keine Regeln für eine Silbentrennung gelernt, noch in einer Textverarbeitung implementiert werden.
- Einheitliche Zeichenbreite: Da jedes Zeichen die gleichen Ausmaße hat, gibt es keine Notwendigkeit Verfahren für einen Blocksatz zu implementieren. Proportionalschriften mit Ausrichtungsproblemen oder sich verbreiternde Zeichen bei Fettdruck gibt es nicht.
- Keine Groß-Klein-Schreibung
- Speichereffizienz: Ein mittels chinesischen Schriftzeichen geschriebener Text nimmt im Mittel weniger als 60 % des Speicherplatzes als ein vergleichbarer englischer oder deutscher Text ein. Selbst nach Kompression beider Texte (pkzip) sind chinesische Texte noch 20 % kompakter als englische oder deutsche Texte.

### 3.5.5 Chinesische Systeme

Als Chinesische Systeme bezeichnet man Softwareprodukte, die es ermöglichen chinesische Schrift auf dem Computer zu verwenden. Neben der reinen Darstellung der Schrift ist auch meist die Möglichkeit vorhanden, chinesische Schriftzeichen einzugeben. Man kann folgende Klassen von Systemen unterscheiden.

- Chinesisches Betriebsystem: z.B. chinesisches Windows, ZWDOS, CCDOS, KC, ...
- Spezielle Programme (Editoren):z.B. Nanji-Star, byx-edit, DingDang, chinese Word, chin.TEX / emacs ,....
- Chinesische Systeme, die ein englisches/deutsches Windows-Betriebsystem ergänzen, so daß die chinesische Schrift verarbeitet werden kann: z.B. RichWin [45], Chinese Star [8], TwinBridge [59], Union Way [60], NJWIN [36], ...
- Unicode Systeme: z.B. Java, Windows NT, ... , wobei noch keine Eingabemethoden definiert sind.

# 4 Das Spracherkennungswerkzeug Janus

Das Sprach-zu-Sprach-Übersetzungssystem Janus der Universität Karlsruhe und der Carnegie Mellon University hat zum Ziel, simultan zwei Sprachen aus einem Pool ineinander zu übersetzen. Dabei erfolgt die Übersetzung in den beiden Richtungen in jeweils 3 Schritten:
Spracherkennung → Textübersetzung → Sprachsynthese.
Hier soll nur der erste Teil von Janus, die Spracherkennungskomponente JRTk (Janus Recognition Toolkit), weiter betrachtet werden. Die Bezeichnung Janus bezieht sich im weiteren deshalb nur noch auf die Spracherkennung.

Janus ist ein Werkzeug, welches den Aufbau eines Spracherkenners ermöglicht und dabei besonders die Aspekte der wissenschaftlichen Forschung betont, d.h. es sollen möglichst einfach neue Erkenntnisse und Verfahren integriert werden können. Dazu wurde besonderen Wert auf die folgenden Punkte gelegt:

- Flexibilität: Viele Spracherkennungsarchitekturen sollen realisierbar sein.
- Skript-Steuerung: Alle Prozesse werden durch wenige Skripte einer leistungsfähigen Skriptsprache (Tcl) gesteuert.
- Erweiterbarkeit: Durch Modularität und Objektorientierung sollen Module einfach austauschbar und ergänzbar sein.
- Kontrollierbarkeit: Einsicht in alle wichtigen Datenstrukturen soll möglich sein.
- Grafische Oberfläche: Durch Tcl/Tk können grafische Ein-/Ausgaben realisiert werden.
- Portabilität: Janus soll auf möglichst vielen Plattformen verfügbar sein.

Geschwindigkeits- und Speicherbedarfsaspekte wurden, obwohl sie natürlich auch wichtige Forschungsthemen sind, erst einmal als zweitrangig betrachtet, um sich besser auf die Erkennungsgenauigkeit konzentrieren zu können.

## 4.1 Einführung in Janus

Das Janussystem, momentan in Version 4 aktuell, hat über mehrere Jahre eine stete Fortentwicklung hinter sich. Im heutigen System können hauptsächlich 3 Bereiche unterschieden werden. Das eigentliche Janussystem, das eine Art Klassenbibliothek für die Spracherkennung darstellt, eine Skriptesammlung, die Standardprobleme der Spracherkennung überdeckt und eine Menge von Zusatzwerkzeugen zur Textformatierung, grafischen Ausgabe und dergleichen.
Um Janus möglichst flexibel zu gestalten, wurde die von John K. Ousterhout [38] entwickelte Skriptsprache Tcl als Basis gewählt. Janus verdankt große Teile seiner Flexibilität und andere Eigenschaften dem zugrundeliegenden Tcl/Tk.
Tcl (Tool command language) wurde Ende der 80‘er Jahre von John Ousterhout als Skriptsprache vorgestellt, die einfach in eigene Programme integrierbar ist. Ein weiteres Ziel war es Tcl als eine Art Klebstoff verwenden zu können, die einzelne Programmteile miteinander verbinden sollte. Um dies zu gewährleisten wurde die einfache Erweiterbarkeit ein weiteres Entwurfsziel. Später als Tcl um das Tk (toolkit) erweitert wurde, wurde Tcl/Tk sehr populär, um schnell Prototypen mit einer grafischen Benutzerschnittstelle unter Unix zu erstellen, so daß sich das Paar Tcl/Tk auch als Programmiersprache etablierte. Tk fand ebenfalls Verbreitung als Mittel um Perl um eine grafische Oberfläche zu erweitern. Um 1995 wurde dann damit begonnen Tcl und Tk auf andere Plattformen wie Windows und Macintosh zu portieren. Al-

lerdings wurde das Look&Feel von Tk zuerst nicht an die Standards der anderen Betriebssysteme angepaßt. Dies wurde dann mit der Version 8.0 im Sommer 1997 nachgeholt, die auch einen Just-in-Time Compiler zur Geschwindigkeitssteigerung mit sich brachte. Tcl/Tk wird stetig weiterentwickelt, so daß auch neuere Entwicklungen, wie Web-Integration, Datenbankanbindung und Systemmanagement Eingang gefunden haben. Als besonders wichtig für die professionelle Anwendbarkeit ist die freie Verfügbarkeit des Quellcodes, die eine rasche Fehlerentdeckung und Bereinigung erlaubt, sowie die hervorragende Dokumentation inklusive der Newsgroups.
Da Tcl an sich nicht objektorientiert ist, wurde für Janus eine Art Objektmodell eingeführt. Dieses erlaubt zwar keine Vererbung oder Klassendefinition, kann aber Daten und Methoden in Objekten übersichtlich zusammenfassen. Für die Realisierung wurde dazu der Unkown-Mechanismus von Tcl benutzt, der bei Erkennen eines unbekannten Befehls die Unkown-Prozedur aufruft, welche beliebig überschrieben werden kann. So können beispielsweise die Objektoperatoren „`. :`“ definiert werden. Jedes Objekt von Janus wird auf einen dynamisch erzeugten, gleichlautenden Befehl in Tcl abgebildet, so daß Tcl weiß, was es mit dem Objekt anzufangen hat.

Bsp.:

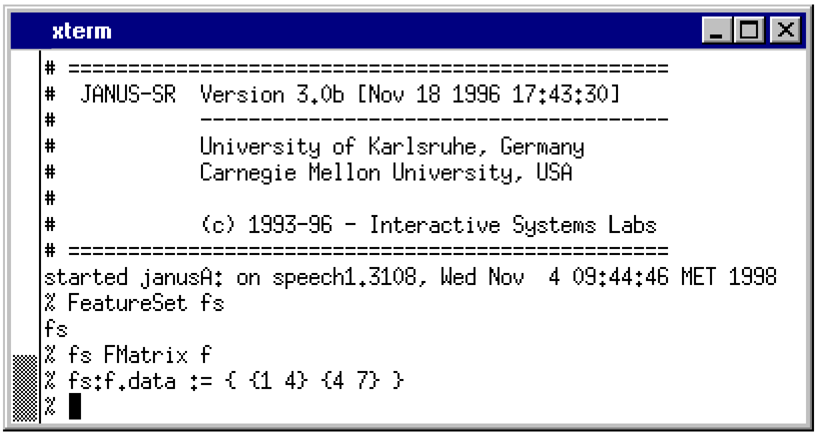
| | |
|---|---|
| `FMatrix m` | (erzeugt Objekt m der Klasse FMatrix) |
| `m := { {1 3} {2 4} }` | (führt Methode := mit Parameter { {1 3} {2 4} } aus) |
| `FeatureSet fs` | (erzeugt Objekt fs der Klasse FeatureSet) |
| `fs FMatrix f` | (erzeugt Objekt f der Klasse FMatrix als Inhalt von fs) |
| `fs : f . data := m` | (dem Unterobjekt data von fs:f wird m zugewiesen) |

Instanzvariablen, welche der Konfiguration von Objekten dienen, werden mit der Methode `configure` angezeigt und können mit ihr auch geändert werden.

Bsp.:

| | |
|---|---|
| `FeatureSet fs` | |
| `fs configure` | (zeigt Konfigurationsvariablen mit Werten) |
| `fs configure -frameShift` | |
| `-> 10.000000` | |
| `fs configure -frameShift 10` | (setzt frameShift auf 10) |

Janus bietet dem Benutzer eine Shell an, in der dieser Systemkommandos, Tcl-Befehle oder Tcl-Skripte ausführen kann. Alternativ kann Janus auch direkt ein Skript übergeben werden (Aufruf: janus Skript.tcl). Die Datei `.janusrc` im Home-Verzeichnis wird beim Start von Janus immer automatisch als erstes ausgeführt. Sie enthält z.B. Pfadangaben für Bibliotheken.



```
# ================================================
#  JANUS-SR  Version 3.0b [Nov 18 1996 17:43:30]
#            -------------------------------------
#            University of Karlsruhe, Germany
#            Carnegie Mellon University, USA
#
#            (c) 1993-96 - Interactive Systems Labs
# ================================================
started janusA: on speech1.3108, Wed Nov  4 09:44:46 MET 1998
% FeatureSet fs
fs
% fs FMatrix f
% fs:f.data := { {1 4} {4 7} }
% █
```

Abbildung 4.1: Janus-Shell in einem X-Window unter Windows

Um alle definierten Klassen von Janus anzuzeigen, wurde der Befehl `type` implementiert. Informationen über Methoden einer Klasse oder eines Objektes können durch die Methode `?` oder `-help` erhalten werden. Der Inhalt von Objekten wird durch die Methode `puts`, die allen Objekten gemein ist, angezeigt. Alternativ kann auch nur der Name des Objektes angegeben werden. Eine Form von Persistenz wird durch die Methoden `load/save` erreicht, die die wichtigsten Objekte bereitstellen.

Um Aufgaben der Spracherkennung ausführen zu können, müssen die entsprechenden Objekte erzeugt und initialisiert werden. Dabei stehen die einzelnen Objekte in einem Abhängigkeitsverhältnis. Das heißt, um manche Objekte erzeugen zu können, müssen benötigte Objekte schon vorhanden sein. Die folgende Abbildung beschreibt die wichtigsten Objekte und gibt die Struktur ihrer Abhängigkeiten wieder.

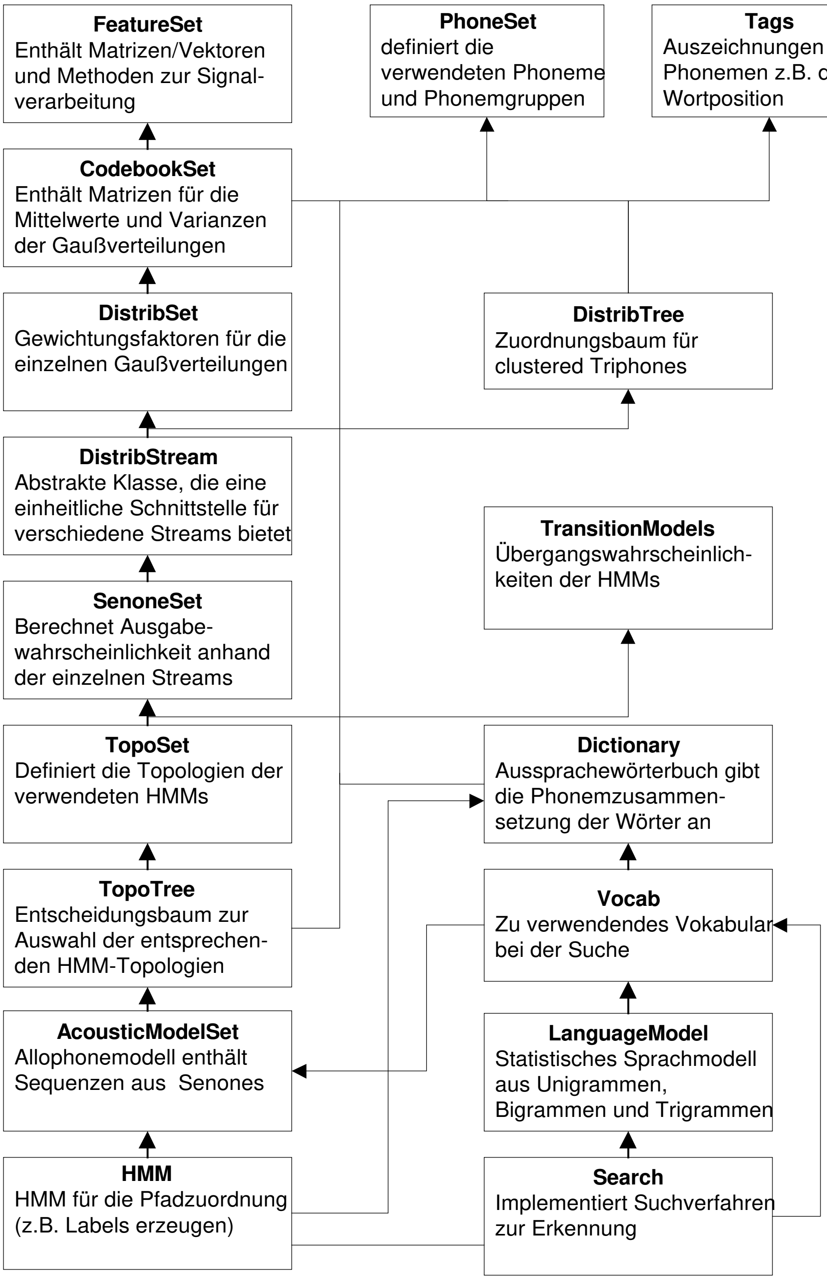

**FeatureSet**
Enthält Matrizen/Vektoren und Methoden zur Signalverarbeitung
**PhoneSet**
definiert die verwendeten Phoneme und Phonemgruppen
**Tags**
Auszeichnungen von Phonemen z.B. die Wortposition
**CodebookSet**
Enthält Matrizen für die Mittelwerte und Varianzen der Gaußverteilungen
**DistribSet**
Gewichtungsfaktoren für die einzelnen Gaußverteilungen
**DistribTree**
Zuordnungsbaum für clustered Triphones
**DistribStream**
Abstrakte Klasse, die eine einheitliche Schnittstelle für verschiedene Streams bietet
**TransitionModels**
Übergangswahrscheinlichkeiten der HMMs
**SenoneSet**
Berechnet Ausgabewahrscheinlichkeit anhand der einzelnen Streams
**TopoSet**
Definiert die Topologien der verwendeten HMMs
**Dictionary**
Aussprachewörterbuch gibt die Phonemzusammensetzung der Wörter an
**TopoTree**
Entscheidungsbaum zur Auswahl der entsprechenden HMM-Topologien
**Vocab**
Zu verwendendes Vokabular bei der Suche
**AcousticModelSet**
Allophonemodell enthält Sequenzen aus Senones
**LanguageModel**
Statistisches Sprachmodell aus Unigrammen, Bigrammen und Trigrammen
**HMM**
HMM für die Pfadzuordnung (z.B. Labels erzeugen)
**Search**
Implementiert Suchverfahren zur Erkennung

Abbildung 4.2: Janusobjekte und deren Abhängigkeiten

Die einzelnen Klassen, die als Menge (Set) ausgebildet sind, enthalten Elemente des entsprechenden Typs. So kann das FeatureSet z.B. mehrere Features enthalten.
Für jeden Featurevektor eines Features kann ein Übereinstimmungsmaß (Score) zu jeder gelernten Repräsentation einer Klasse bestimmt werden, wobei die Repräsentationen bei Janus eine Linearkombination von n-dimensionalen, parameterisierten Gaußverteilungen sind. Dabei bezeichnet n die Dimension des Featureraumes. Die freien, zu lernenden Parameter dieser Repräsentation sind die Mittelwerte und Varianzen (CodebookSet) der Gaußverteilungen, sowie die Gewichtsfaktoren der Linearkombinationen (DistributionSet). Wenn diese multimodalen Gaußverteilungen bekannt sind, können je Frame und Klasse die klassenbedingten Wahrscheinlichkeiten folgendermaßen berechnet werden:

$$p(x \mid s) = \sum_{i=1}^{n_s} c_{s,i} \frac{1}{\sqrt{(2\pi)^d \cdot |\Sigma_{s,i}|}} \cdot e^{-\frac{1}{2}(x-\mu_{s,i})^T \Sigma_{s,i}^{-1} (x-\mu_{s,i})}$$

Um Rundungsfehler bei der Multiplikation zu vermeiden, wird die logarithmierte, klassenbedingte Wahrscheinlichkeit als der Score der Übereinstimmung festgelegt, so daß Multiplikationen in Summationen übergehen. Janus ist so ausgelegt, daß verschiedene Arten von Ausgabewahrscheinlichkeiten benutzt oder diese sogar kombiniert werden können. Das SenoneSet ist für die Scorekombination aus den einzelnen Distribstreams verantwortlich. Neben den Ausgabewahrscheinlichkeiten benötigt das HMM-Objekt die Zustandsübergangswahrscheinlichkeiten (TransitionModels), sowie die gewünschte Topologie (TopoTree/TopoSet). In die Viterbisuche gehen neben dem Dictionary und PhonemSet/Tags auch noch das Languagemodel mit dem Vokabular ein.

Im Anhang B werden alle Janusklassen mit ihren Methoden und deren Kurzbeschreibungen aufgelistet.

## 4.2 Portierung nach Windows

Janus basiert auf den mittels gnu C erweiterten Libraries von Tcl/Tk und wurde auf die institutseigenen Unix-Systeme angepaßt, indem alle Unix-Spezifika an einer Stelle zusammengefaßt wurden. Es existieren somit Varianten für Sun Solaris, Dec Alpha, HP-Unix und Linux auf Intel Prozessoren. Dies deckt zwar einen großen Teil der Unix Betriebssysteme ab, erschwert es aber, Spracherkennungslösungen einzusetzen oder zu entwickeln, da ca. 90% der Anwender mit Windowsbetriebsystemen arbeiten. Auch für die Entwicklung von Spracherkennungslösungen bietet die Windowsplattform entscheidende Vorteile, da bessere Entwicklungsumgebungen für Programmierer, sowie einheitlichere Schnittstellen für Grafik und Audio zur Verfügung stehen. Viele ehemalige reine Unix Tools existieren heute auch für Windows oder werden in erster Linie für Windows weiterentwickelt. Neben diesen Überlegungen der breiten Verfügbarkeit von Windows, spielte die vergleichsweise günstig zu bekommende Rechenleistung bei PCs eine wesentliche Rolle, da Spracherkennung einen sehr hohen Bedarf an Rechenleistung und Speicher hat.

Die Portierung nach Windows war schwieriger und zeitaufwendiger als erwartet. Dies lag zum einen an der Komplexität von Janus und dem damit verbundenen schwierigen Auffinden von Fehlern, und zum anderen an Dokumentationslücken des Quellcodes, die eine unnötig lange Einarbeitungsphase erzwangen.

Die nachfolgende Aufzählung gibt einige Probleme der Portierung wieder:

- Anderes Prozeß-Handling
- Unterschiede in den Dateisystemen
- Unterschiede in den verwendeten Libraries der Compiler
- Verschiedene Zeilenendezeichen
- Verschiedenartiges Speichermanagement
- Leichte Abweichungen in Tcl/Tk
- Compilereigenheiten, die unterschiedliche Resultate liefern !
- ....

Das am Ende doch noch eine stabile Windows Version von Janus zustande kam, liegt nicht unwesentlich an dem exzellenten Debugger aus Visual Studio 5.0.

## 4.3 Benutzerinterface

Durch die Flexibilität von Janus entsteht mitunter auch eine gewisse Komplexität, speziell wenn mehrere Erkennungssysteme gleichzeitig entwickelt werden. Dies führt oft zu Fehlbedienungen von Janus oder auf ein Konzentrieren auf Nebensächlichkeiten, die vom eigentlichen Kern der Spracherkennung ablenken. Durch eine konsequente Benutzerführung, die keine Einschränkungen der Flexibilität von Janus vornimmt, könnte dem abgeholfen werden. Anfängern würde die Einarbeitung wesentlich erleichtert und Experten von Routinearbeiten befreit. Dies war das Ziel beim Entwurf einer grafischen Benutzeroberfläche für Janus, welches es galt nie vollständig aus den Augen zu verlieren.

Als geeignet erschien hierzu eine integrierte Entwicklungsumgebung aus der sich alle Aufgaben heraus komfortabel erledigen lassen. Dazu wurden alle zum Entwurf eines Spracherkenners benötigten Werkzeuge in der grafischen Benutzerschnittstelle JanusUI (Janus User Interface) (Abb. 4.3) zusammen gefaßt, welche noch um Verwaltungsfunktionen und andere Hilfsmittel ergänzt wurde. Als Entwicklungssystem wurde eine 4GL-Sprache unter Windows gewählt, da zur damaligen Zeit Tcl/Tk nur in Version 7.6/4.2 vorlag, so daß es aufgrund seiner noch nicht ausreichenden Anpassung an Windows und Performanceproblemen in der Fensterdarstellung nicht in Betracht kam. Diese Probleme sind in der Tcl/Tk Version 8.0 nun behoben.

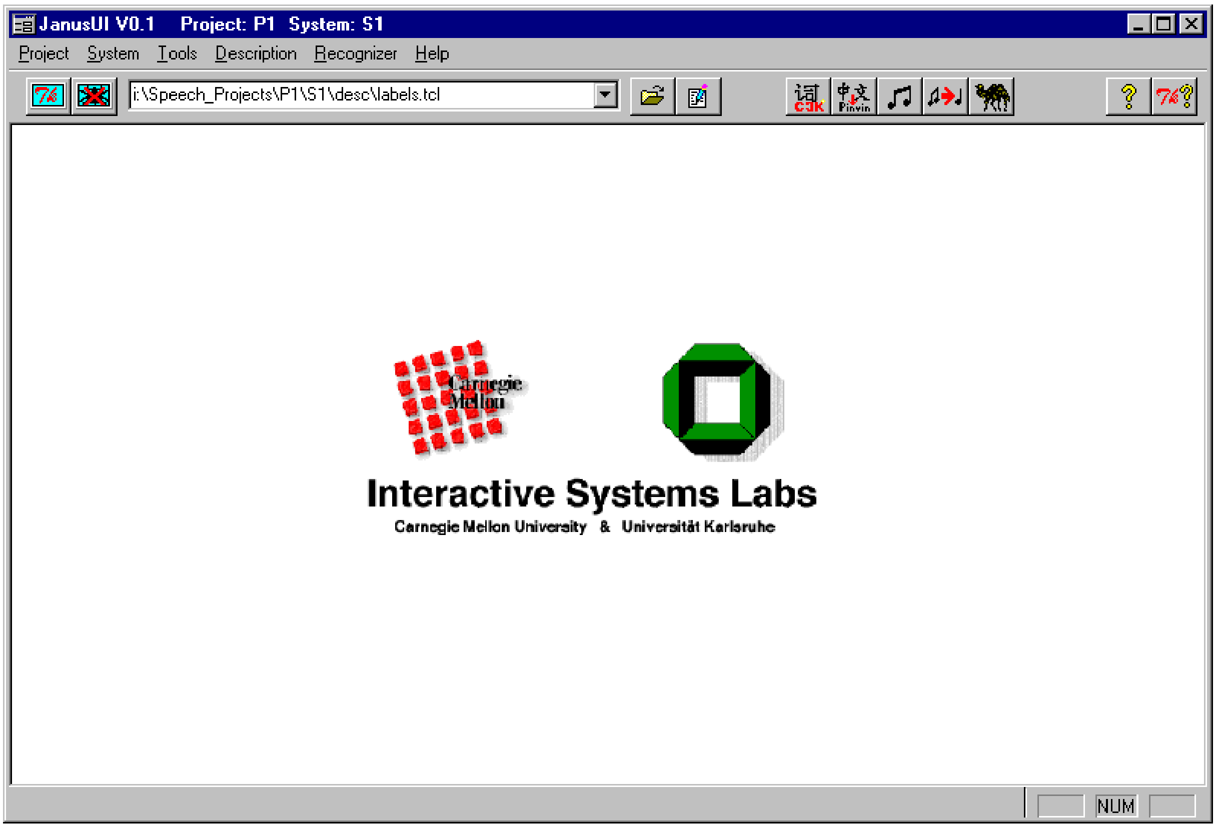

JanusUI V0.1 Project: P1 System: S1
Project
System
Tools
Description
Recognizer
Help
i:\Speech_Projects\P1\S1\desc\labels.tcl
Carnegie Mellon
Interactive Systems Labs
Carnegie Mellon University & Universität Karlsruhe
NUM

Abbildung 4.3: JanusUI – grafische Benutzeroberfläche für Janus

### 4.3.1 Projektverwaltung

Unter den Menüpunkten Project und System befindet sich eine zweistufige Projektverwaltung, die ein einfaches Wechseln zwischen unterschiedlichen Projekten und Systemen ermöglicht. Unter einem System wird dabei ein Spracherkenner verstanden, der sich von anderen nur in der Akustik unterscheidet, aber auf gemeinsame Ressourcen wie Languagemodel, Wörterbuch, Vokabular und Datenbank zugreift. Ein Projekt faßt diese Ressourcen, einschließlich der zugehörigen Systeme, zusammen. Projekte und Systeme können geöffnet (Open), hinzugefügt (New) oder gelöscht (Delete) werden. Ein Kopieren ist vorgesehen, aber noch nicht implementiert. Das aktuelle Projekt/System wird in der Titelleiste des Hauptfensters angezeigt. Beim Neustart wird automatisch das zuletzt bearbeitete Projekt/System geöffnet. JanusUI benötigt die Angabe eines Projektverzeichnisses unter dem alle Projekte mit ihren Systemen jeweils in einem eigenen Verzeichnis abgelegt werden können. Beim Anlegen eines Projektes werden dann die Verzeichnisse "dbase lm dict speech-data" generiert, und beim Erzeugen eines System die Verzeichnisse "init desc train test" mit den entsprechenden Skriptvorlagen (Templates) im Systemverzeichnis erstellt. Die Namen der Verzeichnisse sind frei konfigurierbar.

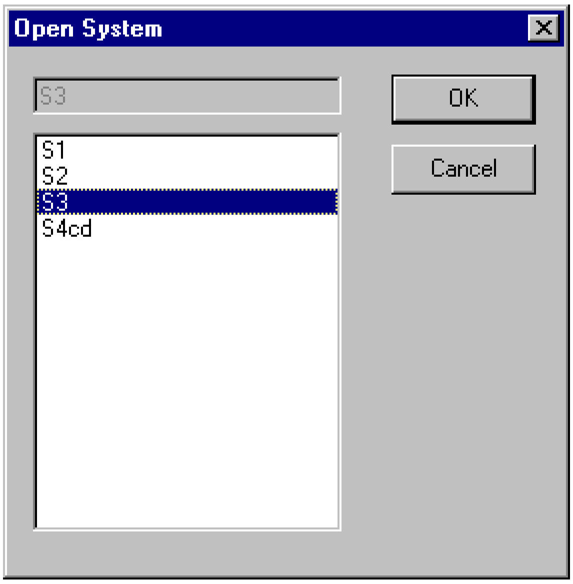
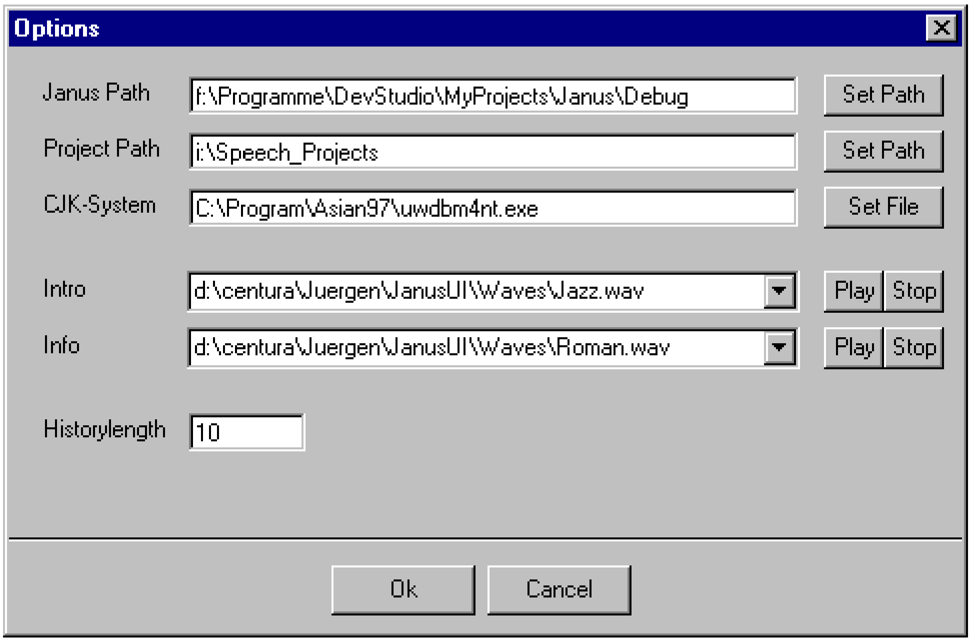


Abbildung 4.4: JanusUI Dialogbox zum Öffnen eines Systems

Die wichtigsten Pfadeinstellungen erfolgen unter dem Menüpunkt Project/Options (Abb. 4.5). Der Januspfad verweist auf das Verzeichnis der portierten Janusversion. Der Projektpfad ist das Standardverzeichnis, in das alle Projekte abgelegt werden. Der Pfad für das CJK-System (asiatische Sprachunterstützung für *C*hinesisch, *J*apanisch, *K*oreanisch) gibt an, wo sich die Unterstützung für die Darstellung und Eingabe für zeichenbasierte Sprachen befindet. In diesem Fall wird hierzu die AsianSuite 97 von Unionway verwendet, welche die Zeichendarstellung, sowie die Eingabemethoden in deutsche oder englische Windowsversionen integriert. Unterstützt werden Chinesisch, Japanisch und Koreanisch, die drei wichtigsten Doppelbytezeichensätze, in jeweils verschiedenen Codierungen.
Bei Intro und Info lassen sich Sounddateien für gewisse Ereignisse auswählen. Historylength gibt die Anzahl der in der Combobox von JanusUI zu speichernden Skripte an.

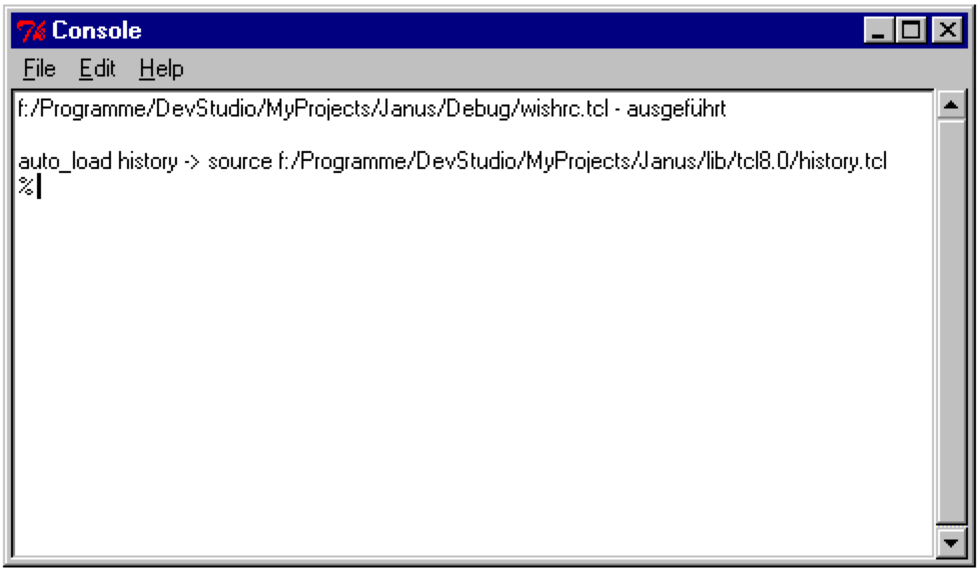


Janus Path f:\Programme\DevStudio\MyProjects\Janus\Debug
Project Path i:\Speech_Projects
CJK-System C:\Program\Asian97\uwdbm4nt.exe
Intro d:\centura\Juergen\JanusUI\Waves\Jazz.wav
Info d:\centura\Juergen\JanusUI\Waves\Roman.wav
Historylength 10

Abbildung 4.5: JanusUI Dialogbox zum Einstellen von Parametern

### 4.3.2 Skripte ausführen und editieren

Unter dem Menüpunkt Projects/Console ist (Abb. 4.6) eine Janusshell zugänglich. Diese hat eine der unter Unix vorhandenen Janus-Shell vergleichbare Funktionalität, wie Cut&Paste, Befehlshistory und einen beliebig großen Scrollbereich. Aus dieser Console heraus kann Janus, wie unter Unix, weiterhin bedient werden.


Console
File
Edit
Help

```
f:/Programme/DevStudio/MyProjects/Janus/Debug/wishrc.tcl - ausgeführt

auto_load history -> source f:/Programme/DevStudio/MyProjects/Janus/lib/tcl8.0/history.tcl
%
```

Abbildung 4.6: Janus Console

Die JanusUI-Symbolleiste (Toolbar, Abb. 4.7) eignet sich zum schnellen Zugang von Funktionen. Sie befindet sich noch im Wandel. In der Abbildung 4.7 sind die momentan verfügbaren Funktionen beschrieben. Besonders angenehm bemerkbar macht sich die History-Funktion der Combobox, in der die zuletzt verwendeten Skripte oder Konfigurationsdateien immer nur einen Mausklick zum Editieren oder Ausführen entfernt sind. So kann oftmals eine Suche vermieden werden. Als Editor wird Notepad von Windows verwendet. Es wären aber auch andere Editoren wie der Programmer's File Editor möglich.

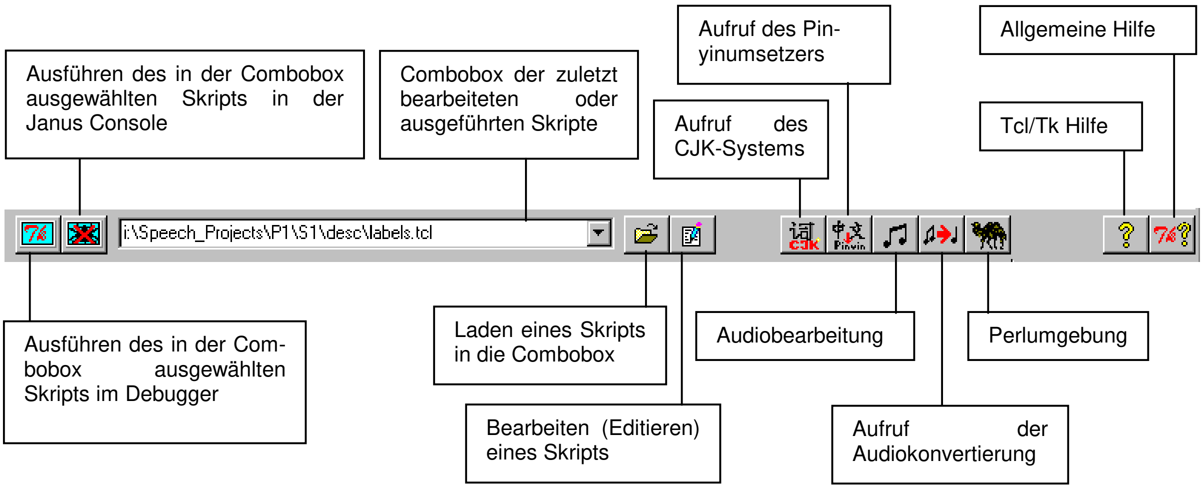

Aufruf des Pinyinumsetzers
Allgemeine Hilfe
Ausführen des in der Combobox ausgewählten Skripts in der Janus Console
Combobox der zuletzt bearbeiteten oder ausgeführten Skripte
Aufruf des CJK-Systems
Tcl/Tk Hilfe
i:\Speech_Projects\P1\S1\desc\labels.tcl
Ausführen des in der Combobox ausgewählten Skripts im Debugger
Laden eines Skripts in die Combobox
Audiobearbeitung
Perlumgebung
Bearbeiten (Editieren) eines Skripts
Aufruf der Audiokonvertierung

Abbildung 4.7: JanusUI Symbolleiste

### 4.3.3 Erzeugen der Konfigurationsdateien für Janus

Um unter Janus einen Spracherkenner aufzubauen, muß zuerst eine Beschreibungsdatei mit Konstruktionsparametern editiert werden und eine Anzahl von Konfigurationsdateien mit anderen benötigten Daten erzeugt werden. Um diesen Erzeugungsprozeß übersichtlich zu gestalten, wurden unter dem Menüpunkt Description die einzelnen erforderlichen Schritte in Gruppen separiert zusammengefaßt. Sie können dort von oben nach unten oder auch in einer anderen beliebigen Reihenfolge bearbeitet werden. Nach Bearbeitung eines Punktes erscheint der entsprechende Menüpunkt dann mit einem Haken, so daß immer der momentane Entwicklungsstand des Systems erfaßt werden kann (Abb. 4.8).

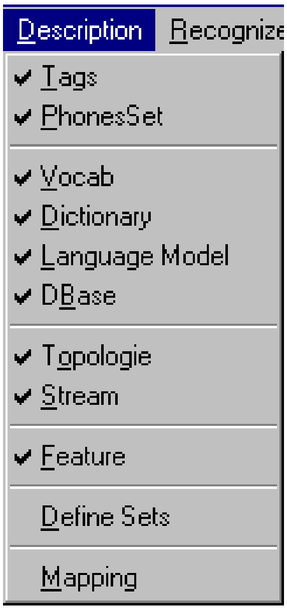
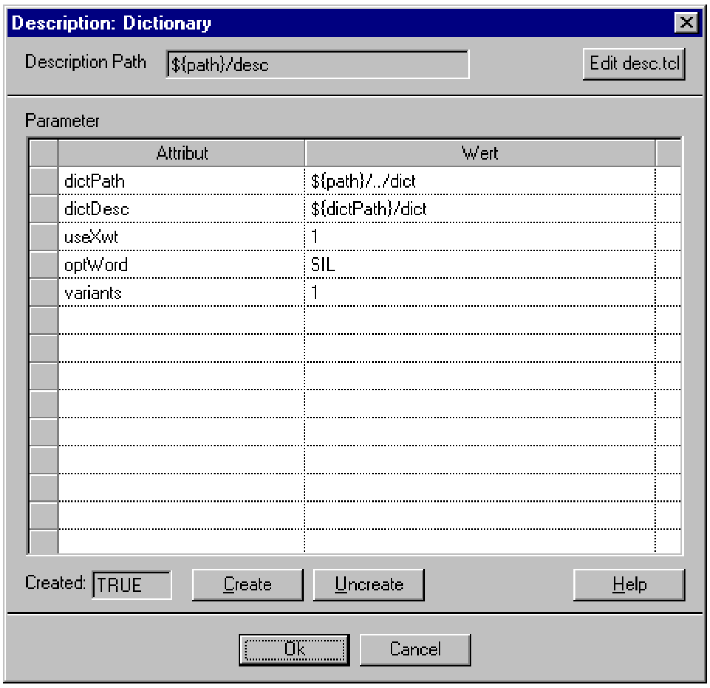


Abbildung 4.9: Description-Menü


Description: Dictionary

Description Path ${path}/desc

Parameter

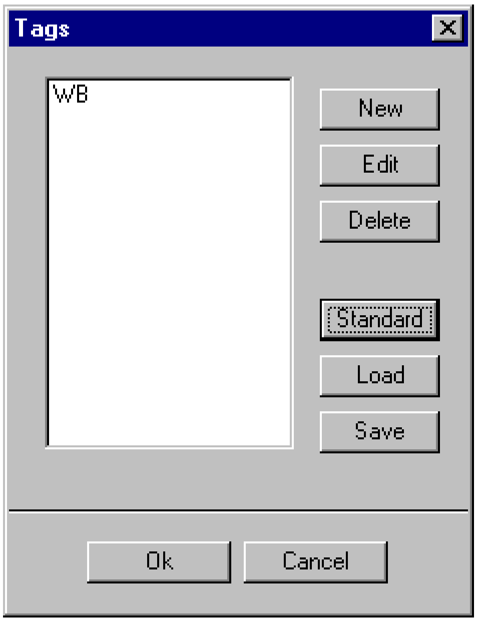
| Attribut | Wert |
|---|---|
| dictPath | ${path}/../dict |
| dictDesc | ${dictPath}/dict |
| useXwt | 1 |
| optWord | SIL |
| variants | 1 |

Created: TRUE

Abbildung 4.8: Dialog zur Parametereingabe

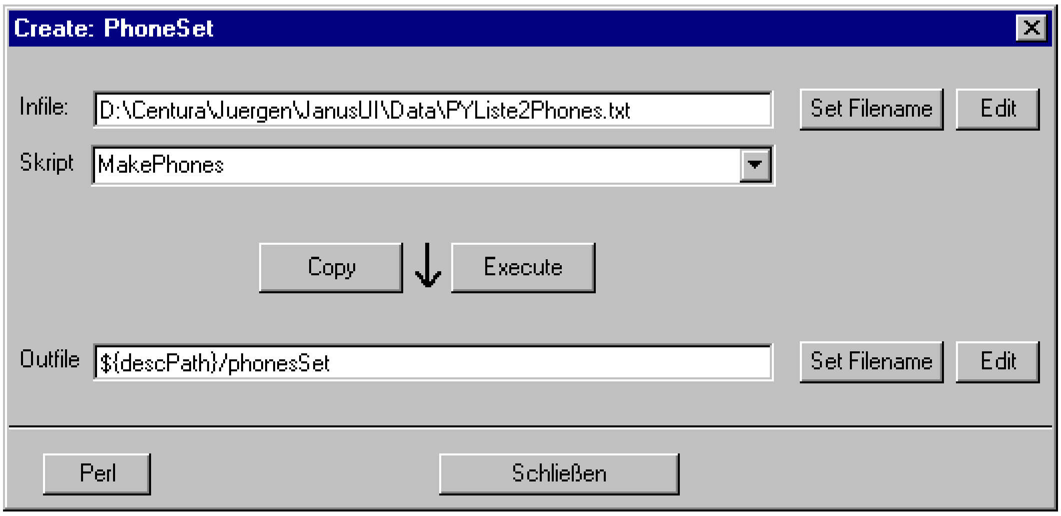
Nach Auswahl eines Menüpunktes erscheint ein Dialog, in dem alle diesen Punkt betreffenden Parameter definiert werden können. Änderungen in diesem Dialog bewirken nach Schließen der Maske eine automatische Änderung der entsprechenden Parameter in der „desc.tcl“-Datei. Die Zuordnung der einzelnen Parameter der „desc.tcl“-Datei zu den entsprechenden Menüpunkten erfolgt dabei über spezielle, als Kommentare geschützte Auszeichnungen, so daß bei Änderung des „desc.tcl“-Templates auch die Attribut-Wert-Paare automatisch angepaßt werden. Über den Button „Edit desc.tcl“ kann die „desc.tcl“-Datei auch von Hand editiert werden. Im Created-Feld erscheint, gemäß dem Haken im Menü, ob die entsprechenden Konfigurationsdateien schon erzeugt wurden. Diese können mit „Create“ erzeugt bzw. mit „Uncreate“ wieder gelöscht werden, wobei Uncreate allerdings nur das Created-Flag zurücksetzt.

Im folgenden werden die nach dem Betätigen von „Create“ erscheinenden Dialogmasken, die das Erzeugen der entsprechenden Dateien steuern, kurz erläutert.



Tags zeichnen Phoneme aus. So können Tags z.B. Phoneme unterscheiden, die am Anfang, in der Mitte oder am Ende eines Wortes vorkommen. Die Namen von Tags können beliebig vergeben werden. Wichtig ist nur, daß sie später konsistent eingesetzt werden.
Der nebenstehende Dialog (Abb. 4.10) erlaubt eine Liste von Tags zu erstellen. Mit „New“ oder „Delete“ werden neue Tags hinzugefügt bzw. gelöscht. „Edit“ erlaubt den Namen vorhandener Tags zu verändern. „Standard“ liefert eine Standard-Tag-Liste. „Load“ und „Save“ erlauben es Listen zu importieren oder zu exportieren. Mittels „Ok“ wird die sich in der Listbox befindende Liste ins aktuelle System übernommen.

Abbildung 4.10: Erzeugung der Tags-Datei

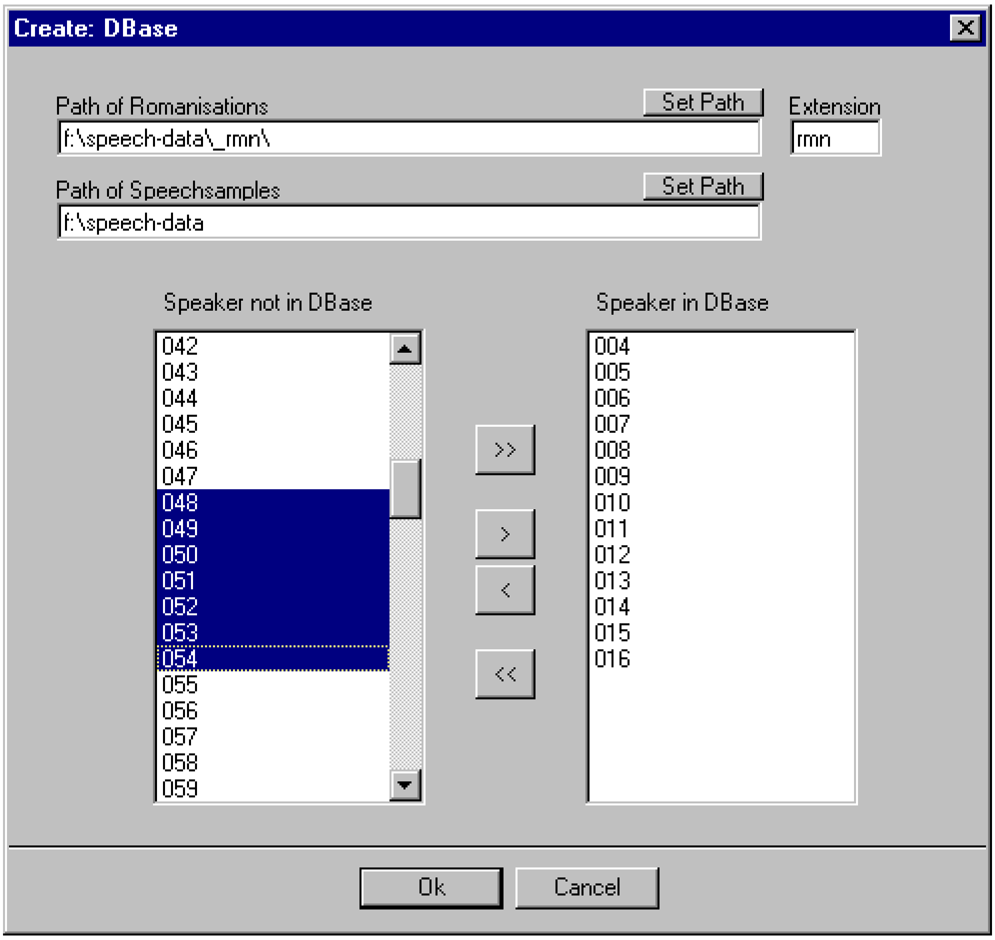


Infile: D:\Centura\Juergen\JanusUI\Data\PYListe2Phones.txt
Skript: MakePhones
Outfile: ${descPath}/phonesSet

Abbildung 4.11: Erzeugung des PhonesSets

Dieser Dialog (Abb. 4.11) erlaubt das Erzeugen eines PhonesSets. Dies kann entweder durch simples Kopieren (Copy) einer externen Datei (Infile) ins System (Outfile) erfolgen, oder basierend auf einem Skript anhand eines Datenfiles (Infile) neu erzeugt werden (Execute). Dazu geeignete Skripte können aus der Combobox „Skript“ ausgewählt werden. Die Skripte stammen dabei aus einem Pool von Perl (oder auch Tcl)-Skripten. Durch „Edit“ können die entsprechenden Dateien mit dem Standardeditor editiert werden. „Set Filename“ erlaubt, wie auch in anderen Dialogen, das bequeme Auswählen der Dateien mittels der Maus anhand des Windows Standarddialogs „Dateiauswahl“. Der Button Perl erlaubt den schnellen Wechsel zur Perlumgebung, um evtl. Skripte anzupassen. Alle Felder, auch in anderen Dialogen, merken sich den zuletzt angezeigten Inhalt und sind somit beim Öffnen bereits mit Standardwerten vorbelegt.

Die Dialogmasken für die Description-Menüpunkte Vocab, Dictionary und Language Model entsprechen der Dialogmaske von PhonesSet. Die Eingabefelder sind den Aufgaben gemäß vorbelegt.



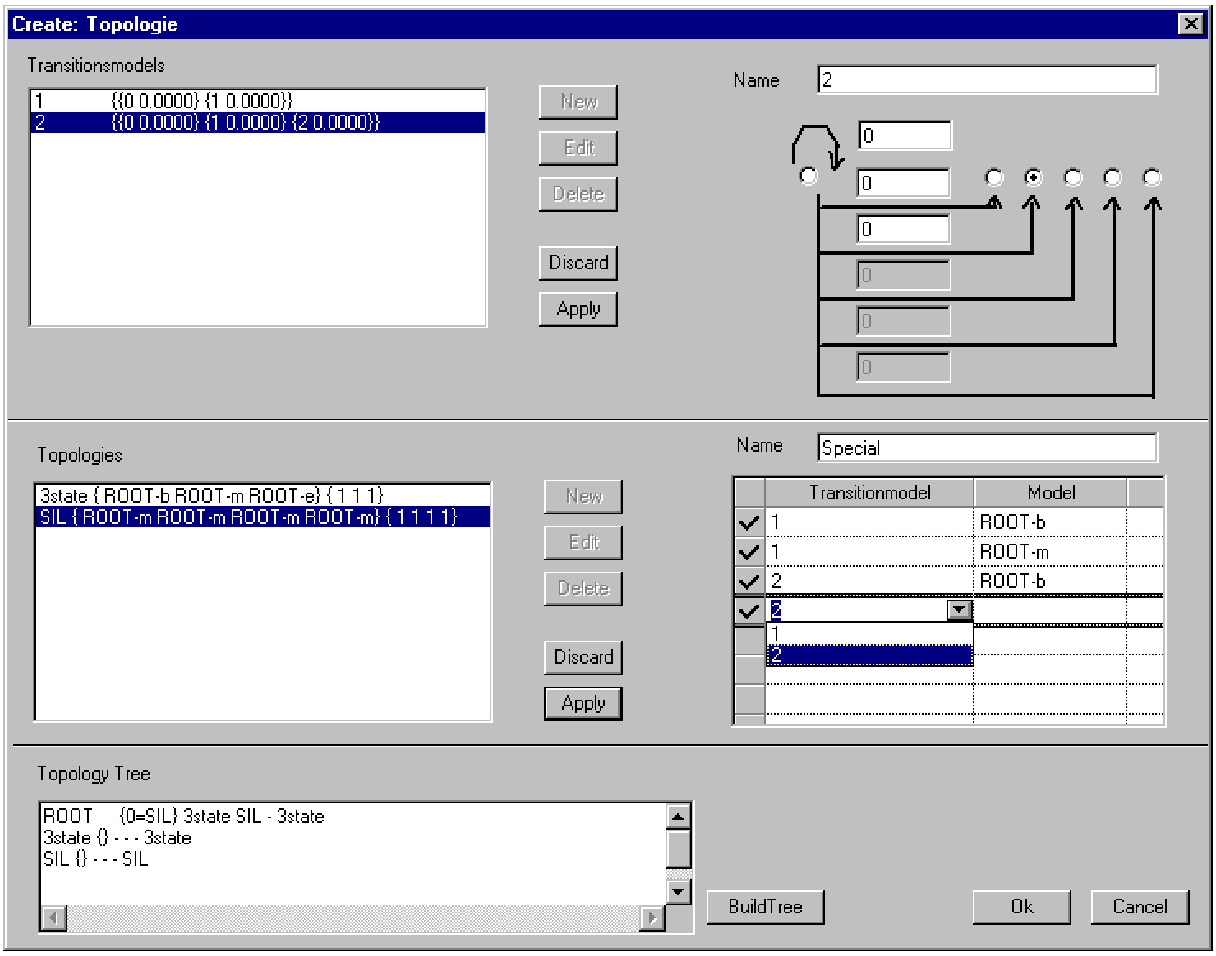
Path of Romanisations: f:\speech-data_rmn\
Extension: rmn
Path of Speechsamples: f:\speech-data

Speaker not in DBase:
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059

Speaker in DBase:
004
005
006
007
008
009
010
011
012
013
014
015
016

Abbildung 4.12: Erzeugung der DBase

Janus benutzt zum Ausführen der meisten Skripte zwei Datenbanken, die Daten über Sprecher und deren Äußerungen enthalten. Diese Datenbanken sind als asoziative Tcl-Arrays realisiert. Sie können über die oben abgebildete Maske erzeugt werden. Dazu werden der Pfad der Transkriptionen oder der Pfad der Texte in Lautschriftumschrift und der Pfad der Audiofiles benötigt. In der linken Listbox erscheinen dann alle möglichen Sprecher, d.h. Sprecher die sowohl Audiofiles als auch zugehörige Transkriptionen besitzen. Diese können dann durch die Buttons (alle nach links, ausgewählte nach links, ausgewählte nach rechts, alle nach rechts) zwischen den beiden Listboxen verschoben werden. Aus den Sprechern der rechten Listbox wird dann beim Verlassen des Dialogs automatisch die Datenbank aufgebaut. Da beim erneuten Aufruf die Verteilung der Sprecher erhalten bleibt, kann die Datenbank auch sehr einfach wieder modifiziert werden.



Transitionsmodels:

```
1        {{0 0.0000} {1 0.0000}}
2        {{0 0.0000} {1 0.0000} {2 0.0000}}
```

Name: 2

Transition fields: 0, 0, 0, 0, 0, 0

Topologies:

```
3state { ROOT-b ROOT-m ROOT-e} { 1 1 1}
SIL { ROOT-m ROOT-m ROOT-m ROOT-m} { 1 1 1 1}
```

Name: Special

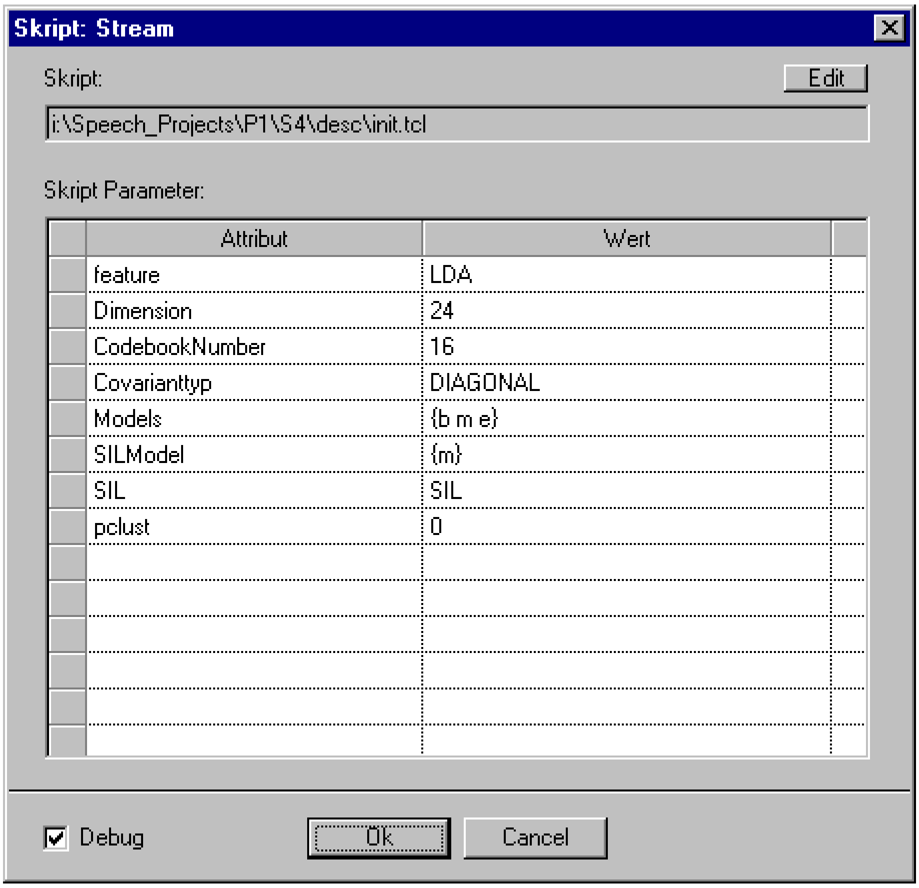
| | Transitionmodel | Model |
|---|---|---|
| ✓ | 1 | ROOT-b |
| ✓ | 1 | ROOT-m |
| ✓ | 2 | ROOT-b |
| ✓ | 2 | |

Topology Tree:

```
ROOT     {0=SIL} 3state SIL - 3state
3state {} - - - 3state
SIL {} - - - SIL
```

Abbildung 4.13: Festlegen der HMM-Topologien

Für die Definition der HMM-Topologien sind die drei Dateien „TransitionModels“, „topologies“, sowie „topologyTree“ zuständig. Sie wurden in einem Dialog thematisch zusammengefaßt.
Jeder Zustand eines HMM entspricht genau einem Transitionmodel. Das Transitionmodel kann als Typ des HMM-Zustandes aufgefaßt werden. Die Festlegungen betreffen den Grad der Vorwärtstopologie, sowie die einzelnen Zustandsübergangswahrscheinlichkeiten. Mittels „New“ und „Delete“ können neue Transitionmodels hinzugefügt bzw. Transitionmodels gelöscht werden. „Edit“ erlaubt das Ändern bestehender Modelle. Mit „Discard“ können Änderungen verworfen oder mit „Apply“ festgeschrieben werden. Das Anlegen neuer Transitionmodels erfolgt in den drei Schritten: Name festlegen, Radio Button für Grad der Vorwärtstopologie wählen, und dann die entsprechenden Zustandsübergangswahrscheinlichkeiten eintragen.
Im mittleren Bereich der Eingabemaske (Abb. 4.13) können neue Topologien, basierend auf den Transitionmodels, definiert werden. Zuerst muß wieder ein Topologiename vergeben werden. Dann können den einzelnen Zuständen dieser Topologie schon definierte Transitionmodels und ein Modelname zugeordnet werden. Für die restlichen Buttons gilt das oben Gesagte.
Im Abschnitt „Topology Tree“ kann mit dem Button „BuildTree“ dann automatisch der Entscheidungsbaum, der den Phonemen das richtige HMM zuordnen soll, aufgebaut werden. Im nebenstehenden Feld „Topology Tree” kann dieser dann überprüft werden.



Skript: i:\Speech_Projects\P1\S4\desc\init.tcl

☑ Debug

| Attribut | Wert |
|---|---|
| feature | LDA |
| Dimension | 24 |
| CodebookNumber | 16 |
| Covarianttyp | DIAGONAL |
| Models | {b m e} |
| SILModel | {m} |
| SIL | SIL |
| pclust | 0 |

Abbildung 4.14: Erzeugen der Stream-Dateien

Unter den Stream-Dateien werden die Konfigurationsdateien verstanden, die die Objekte für die Berechnung der Ausgabewahrscheinlichkeiten des HMM oder NN definieren. Im einzelnen sind dies das „codebookSet“, das „distribSet“, der „distribTree“ und bei kontextabhängigen Systemen zusätzlich das „ptreeSet“. Zum Erzeugen dieser Dateien dient ein Skript (init.tcl) der Janus Skriptesammlung, welches durch obige Dialogbox konfigurierbar ist. Hier können z.B. das zu verwendende Feature, sowie Dimension, Anzahl und Typ der Codebücher angegeben werden, desweiteren, welche Modelle für die Phoneme und das Silence-Phonem verwendet werden soll. Das Init-Skript kann auch direkt editiert oder im Debugmodus ausgeführt werden.

Das Init-Skript erzeugt die Strukturdefinitionen für die oben erwähnten Objekte, indem für jedes Phonem aus dem PhonesSet zu jedem Modell ein Codebook zum CodebookSet und eine Distribution zum DistribSet hinzugefügt wird. Das SIL-Phonem wird gesondert betrachtet, da hierfür eine andere Modelldefinition möglich ist.
Die erzeugte Beschreibungsdatei des DistribTree besteht aus einer Menge von Knoten, die entweder von Typ Wurzel, innerer Knoten oder Blatt sind. Jeder Knoten besteht aus den sechs Feldern: „Namen“, „Frage“, sowie die möglichen Antworten „Ja“, „Nein“, „Weiß nicht“ und ein Feld für die „Blattbezeichnung“, falls es sich um ein Blatt handelt. Nur die inneren Knoten besitzen eine wirkliche Frage, die anderen nur ein Dummyfeld. Der DistribTree hat für jedes Model seine eigene Wurzel (z.B. ROOT-b, ROOT-m, ROOT-e).

Bsp.:

```
ROOT-b       {}      hook--i1-b hook--i1-b hook--i1-b -
hook--i1-b   {0=-i1} hook--i2-b -i1-b - -
-i1-b        {}      - - - -i1-b
.....
```

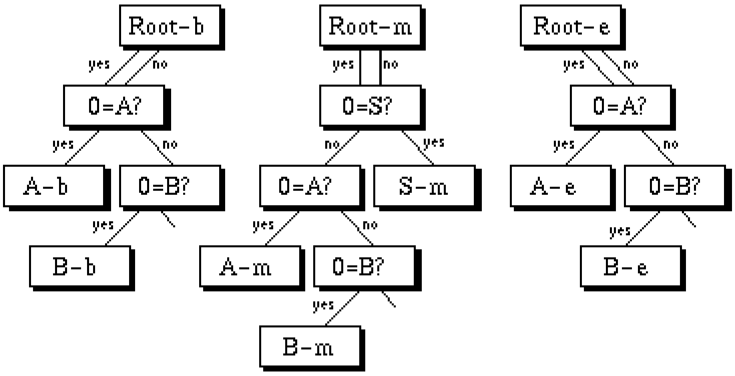
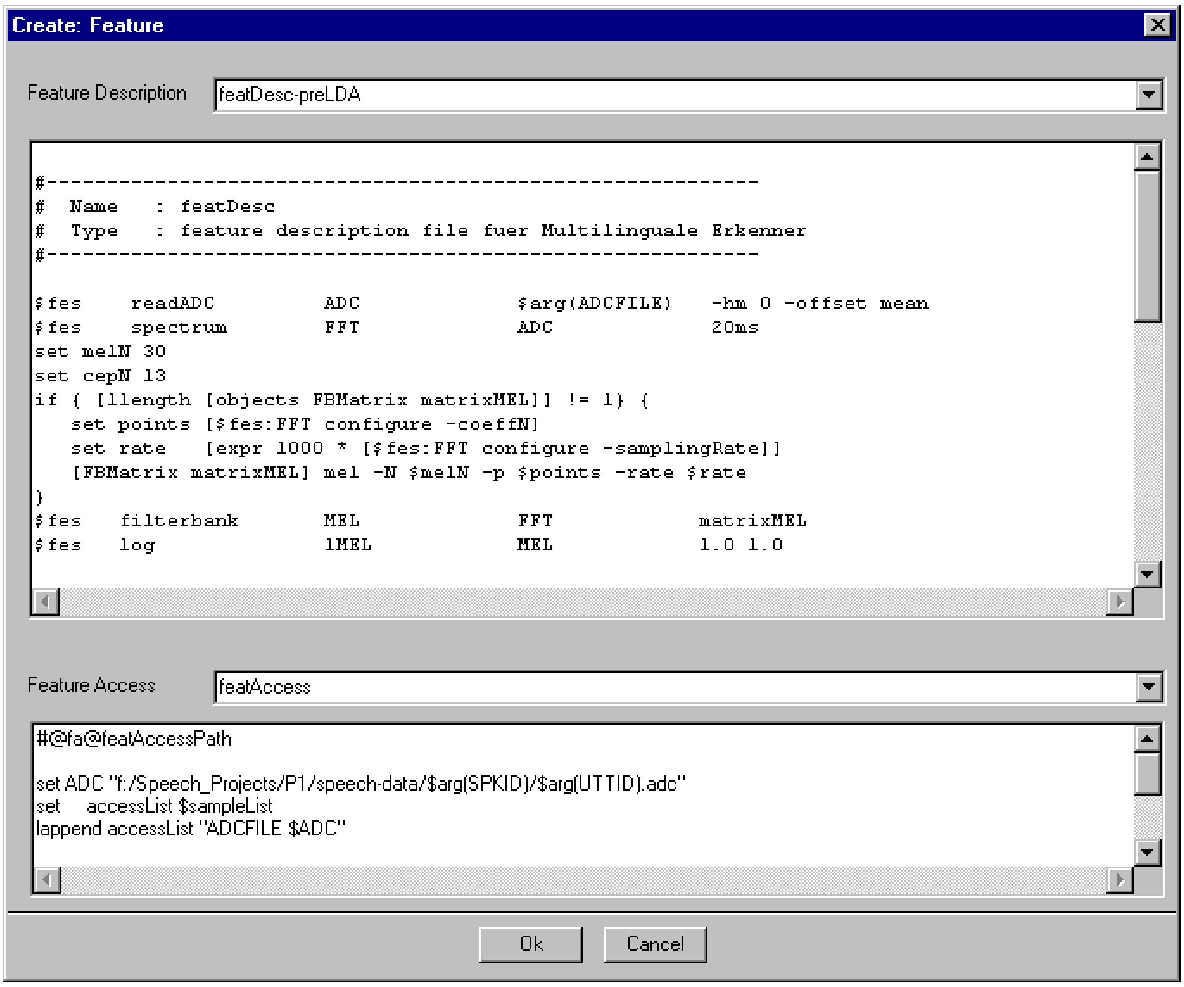

Root-b
yes
no
0=A?
yes
no
A-b
0=B?
yes
B-b
Root-m
yes
no
0=S?
no
yes
0=A?
S-m
yes
no
A-m
0=B?
yes
B-m
Root-e
yes
no
0=A?
yes
no
A-e
0=B?
yes
B-e

Abbildung 4.15: Beispiel DistribTree

Unterhalb des Menüpunkt Description/Feature können die Schritte der Vorverarbeitung, vom Audiosignal zu den Featurevektoren (Merkmalsvektoren), definiert werden (Abb. 4.16).


Create: Feature
Ok
Cancel

Feature Description: featDesc-preLDA

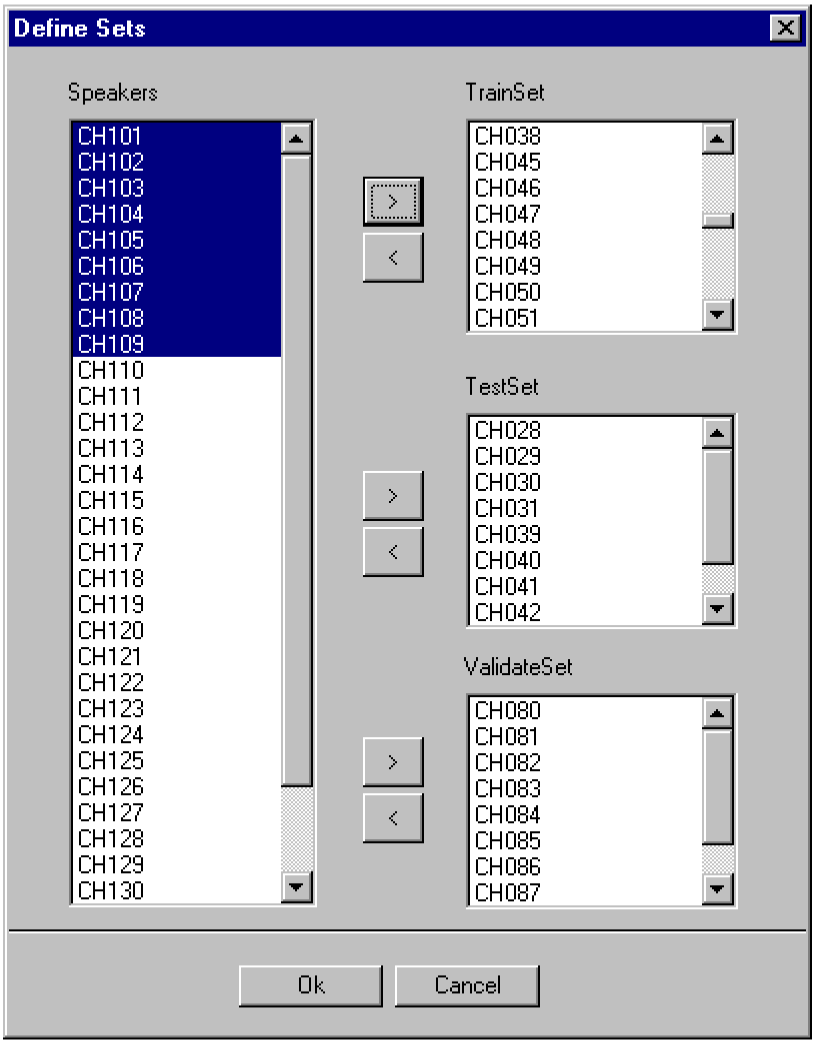
```
#------------------------------------------------------------
#  Name    : featDesc
#  Type    : feature description file fuer Multilinguale Erkenner
#------------------------------------------------------------

$fes    readADC         ADC             $arg(ADCFILE)   -hm 0 -offset mean
$fes    spectrum        FFT             ADC             20ms
set melN 30
set cepN 13
if { [llength [objects FBMatrix matrixMEL]] != 1} {
   set points [$fes:FFT configure -coeffN]
   set rate   [expr 1000 * [$fes:FFT configure -samplingRate]]
   [FBMatrix matrixMEL] mel -N $melN -p $points -rate $rate
}
$fes   filterbank       MEL             FFT            matrixMEL
$fes   log              lMEL            MEL            1.0 1.0
```

Feature Access: featAccess

```
#@fa@featAccessPath

set ADC "f:/Speech_Projects/P1/speech-data/$arg(SPKID)/$arg(UTTID).adc"
set     accessList $sampleList
lappend accessList "ADCFILE $ADC"
```

Abbildung 4.16: Definition der Vorverarbeitung und des Zugriffspfades

Zur Definition der Vorverarbeitung (Feature Description) dient ein Tcl-Skript, welches, wie auch der Zugriffspfad (Feature Access) in der oben gezeigten Maske, editiert werden kann. Die Comboboxen erlauben es schnell zwischen verschiedenen Definitionen umschalten zu können.



Speakers
CH101
CH102
CH103
CH104
CH105
CH106
CH107
CH108
CH109
CH110
CH111
CH112
CH113
CH114
CH115
CH116
CH117
CH118
CH119
CH120
CH121
CH122
CH123
CH124
CH125
CH126
CH127
CH128
CH129
CH130

TrainSet
CH038
CH045
CH046
CH047
CH048
CH049
CH050
CH051

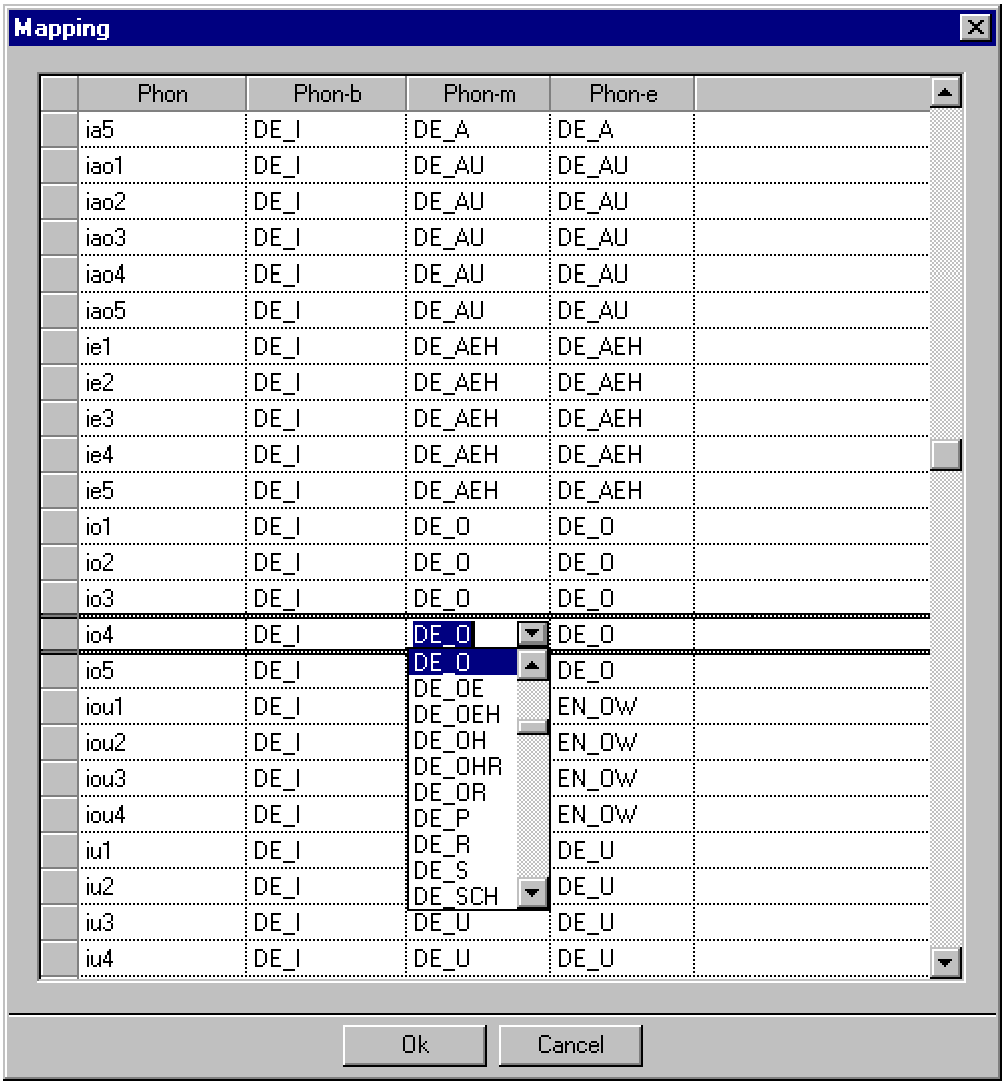
TestSet
CH028
CH029
CH030
CH031
CH039
CH040
CH041
CH042

ValidateSet
CH080
CH081
CH082
CH083
CH084
CH085
CH086
CH087

Abbildung 4.17: Aufteilung der Sprecher in Trainings-, Test-, Evaluationsmenge

Um disjunkte Mengen von Sprechern für das Training, den Test und die Evaluation zu generieren, können die Sprecher aus der Datenbank (linkes Fenster) auf die unterschiedlichen Mengen (rechte Fenster) verteilt werden. Es kann natürlich auch nur ein Teil der Sprecher aus der Datenbank verteilt werden. Die Disjunktheit der Texte muß noch manuell überprüft werden.

Diese strikte Trennung der Daten in Trainings-, Test- und Evaluationsmenge ist nötig, damit die Aussagen von Tests nicht durch unbemerkt mittrainierte Testsachverhalte beeinflußt werden. Dies gilt um so mehr für die Evaluationsmenge, auf der die engültige Performance gemessen wird.



| Phon | Phon-b | Phon-m | Phon-e |
|---|---|---|---|
| ia5 | DE_I | DE_A | DE_A |
| iao1 | DE_I | DE_AU | DE_AU |
| iao2 | DE_I | DE_AU | DE_AU |
| iao3 | DE_I | DE_AU | DE_AU |
| iao4 | DE_I | DE_AU | DE_AU |
| iao5 | DE_I | DE_AU | DE_AU |
| ie1 | DE_I | DE_AEH | DE_AEH |
| ie2 | DE_I | DE_AEH | DE_AEH |
| ie3 | DE_I | DE_AEH | DE_AEH |
| ie4 | DE_I | DE_AEH | DE_AEH |
| ie5 | DE_I | DE_AEH | DE_AEH |
| io1 | DE_I | DE_O | DE_O |
| io2 | DE_I | DE_O | DE_O |
| io3 | DE_I | DE_O | DE_O |
| io4 | DE_I | DE_O | DE_O |
| io5 | DE_I | | DE_O |
| iou1 | DE_I | | EN_OW |
| iou2 | DE_I | | EN_OW |
| iou3 | DE_I | | EN_OW |
| iou4 | DE_I | | EN_OW |
| iu1 | DE_I | | DE_U |
| iu2 | DE_I | | DE_U |
| iu3 | DE_I | DE_U | DE_U |
| iu4 | DE_I | DE_U | DE_U |

Abbildung 4.18: Phonemmapping

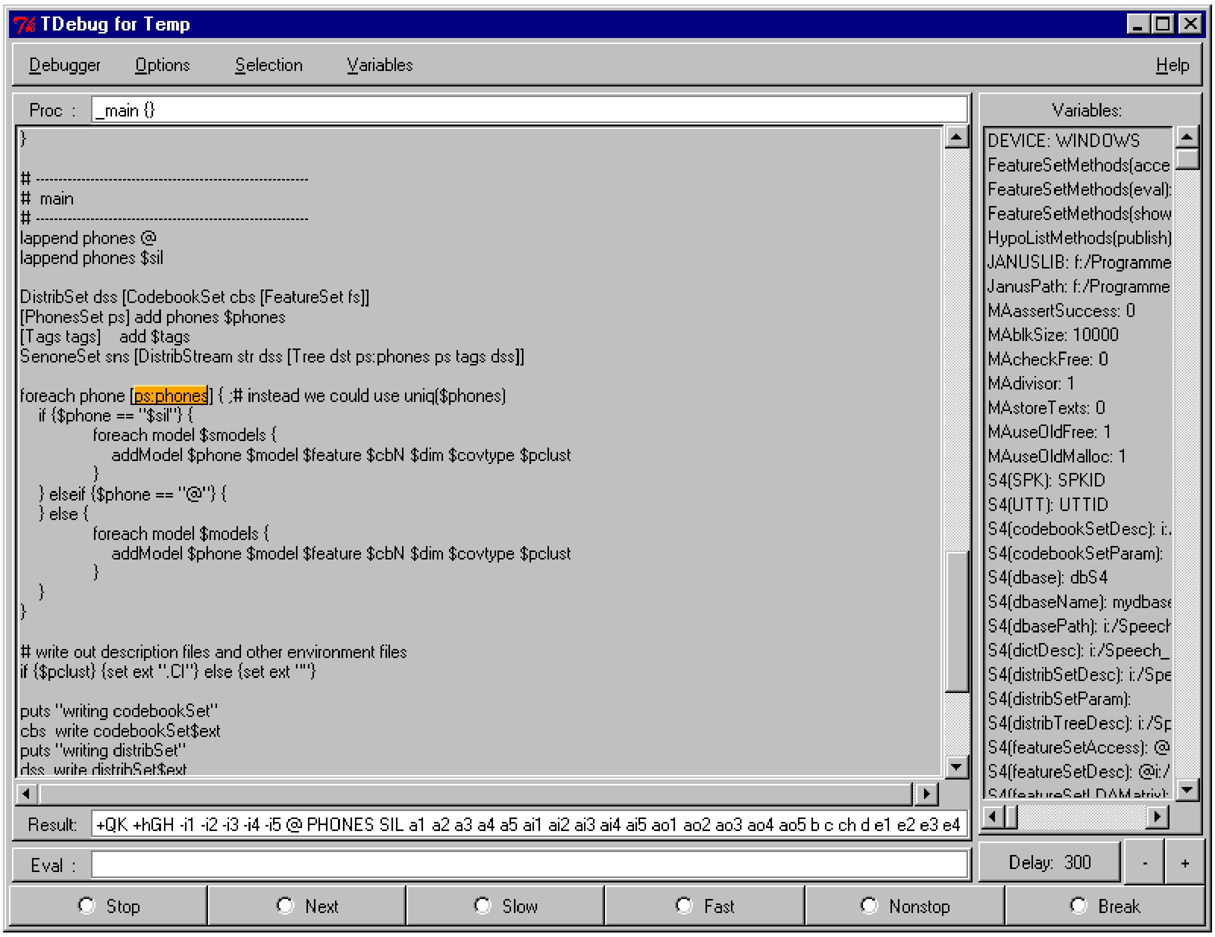
Da beim Aufbau eines neuen Erkenners noch keine Codebücher und Verteilungen vorliegen, müßten diese mit Zufallswerten initialisiert werden. Dies ist zwar möglich, verlängert aber sehr stark den Trainingsablauf. Deshalb nimmt man lieber von einem vorhandenen Erkenner ähnliche Codebücher und Verteilungen und ordnet sie denen des neuen Erkenners zu. Ideal dazu wäre ein Phonemerkenner der internationalen Lautschrift. Der an der Universität Karlsruhe entwickelte multilinguale Erkenner [52], der Phoneme der Sprachen, Deutsch, Englisch, Japanisch und Spanisch enthält, kommt dem schon nahe.
Der Phonemmapping-Dialog erlaubt bequem jedem Phonemmodell des neuen Erkenners ein Phonem des multilingualen Erkenners zuzuordnen. Nach dem Schließen des Dialogs wird die entsprechende Mappingliste generiert, mit der dann das Tcl-Skript „rewrite.tcl“ die Initialisierung vornehmen kann.

### 4.3.4 Skriptausführung

Alle Standard-Skripte aus der Skriptesammlung sind unter dem Menüpunkt Recognizer aufrufbar. Wie bei der Erzeugung der Stream-Dateien erscheint eine Dialogmaske (Abb. 4.14), in der dann Grundparameter eingestellt werden können, bevor das Skript ausgeführt wird. Optional kann die Ausführung auch im Debugger erfolgen.

### 4.3.5 Debugger und hierarchischer Editor für Tcl/Tk

Besonders in der Portierungsphase von Janus war ein Debugger für Tcl unbedingt erforderlich. Aber auch beim späteren Aufbau von Spracherkennern ist ein solcher Debugger ein nützliches Werkzeug. Gregor Schmid bietet mit TDebug Version 1.0 (c) 1993 einen freien Debugger (GNU General Public License) für Tcl an. Da dieser Debugger selbst in Tcl/Tk implementiert ist, ist er portabel und läuft sowohl auf Windows- wie auch Unix-Plattformen. TDebug wurde im Rahmen dieser Arbeit angepaßt, so daß er nun auch problemlos mit Janus-Skripten zusammen funktioniert.



Proc : _main {}

```
}

# ----------------------------------------------------------------
#  main
# ----------------------------------------------------------------
lappend phones @
lappend phones $sil

DistribSet dss [CodebookSet cbs [FeatureSet fs]]
[PhonesSet ps] add phones $phones
[Tags tags]    add $tags
SenoneSet sns [DistribStream str dss [Tree dst ps:phones ps tags dss]]

foreach phone [ps:phones] { ;# instead we could use uniq($phones)
   if {$phone == "$sil"} {
            foreach model $smodels {
               addModel $phone $model $feature $cbN $dim $covtype $pclust
            }
   } elseif {$phone == "@"} {
   } else {
            foreach model $models {
               addModel $phone $model $feature $cbN $dim $covtype $pclust
            }
   }
}

# write out description files and other environment files
if {$pclust} {set ext ".CI"} else {set ext ""}

puts "writing codebookSet"
cbs  write codebookSet$ext
puts "writing distribSet"
dss  write distribSet$ext
```

Result: +QK +hGH -i1 -i2 -i3 -i4 -i5 @ PHONES SIL a1 a2 a3 a4 a5 ai1 ai2 ai3 ai4 ai5 ao1 ao2 ao3 ao4 ao5 b c ch d e1 e2 e3 e4

Eval :

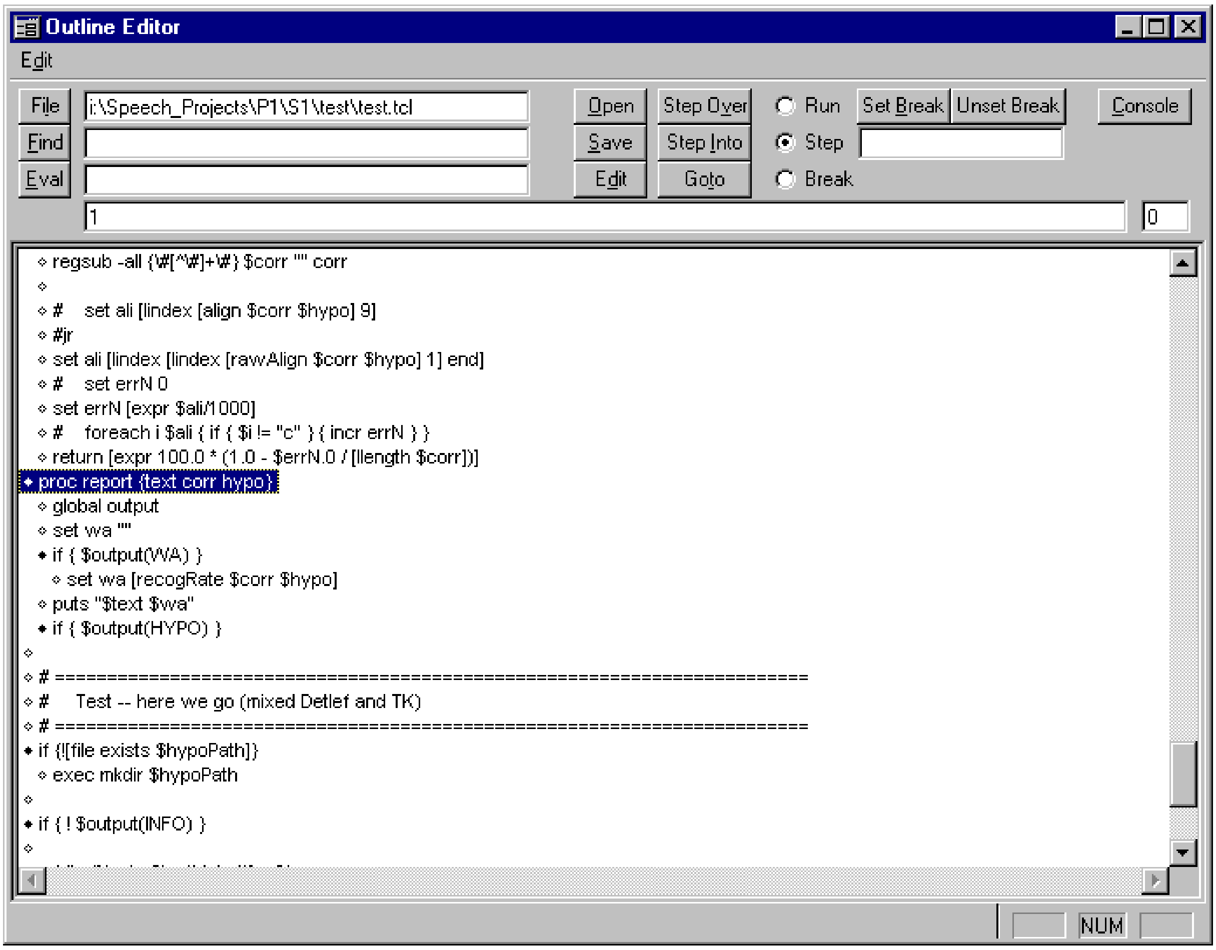
```
DEVICE: WINDOWS
FeatureSetMethods(acce
FeatureSetMethods(eval):
FeatureSetMethods(show
HypoListMethods(publish)
JANUSLIB: f:/Programme
JanusPath: f:/Programme
MAassertSuccess: 0
MAblkSize: 10000
MAcheckFree: 0
MAdivisor: 1
MAstoreTexts: 0
MAuseOldFree: 1
MAuseOldMalloc: 1
S4(SPK): SPKID
S4(UTT): UTTID
S4(codebookSetDesc): i:.
S4(codebookSetParam):
S4(dbase): dbS4
S4(dbaseName): mydbase
S4(dbasePath): i:/Speech
S4(dictDesc): i:/Speech_
S4(distribSetDesc): i:/Spe
S4(distribSetParam):
S4(distribTreeDesc): i:/Sp
S4(featureSetAccess): @
S4(featureSetDesc): @i:/
```

Abbildung 4.19: TDebug

TDebug zeigt im Einzelschrittmodus den gerade bearbeiteten Befehlsteil farbig unterlegt an. Leider ist dies der schon bearbeitete Teil und nicht wie sonst bei Debuggern üblich der nächste zu bearbeitende Teil. Dies vereinfachte zwar die Erstellung des Debuggers wesentlich, da so bedingte Sprünge und Funktionsaufrufe nicht zuvor ausgewertet werden müssen. Dies ist der Übersichtlichkeit aber nicht zuträglich. Neben der Einzelschrittverarbeitung existiert auch die Möglichkeit eines animierten Ablaufes mit einstellbarer Geschwindigkeit. Gesetzte Breakpoints erlauben, an einer bestimmten Stelle wieder in den Einzelschrittverarbeitungsmodus zu wechseln, um in der linken Listbox die aktuelle Belegung der Variablen zu untersuchen. Im Resultfeld erscheint immer das Ergebnis des gerade bearbeiteten Befehlsteils. Im Evalfeld kann zu jedem Zeitpunkt ein beliebiger zu verarbeitender Befehl eingegeben werden.

Obwohl TDebug sehr gut programmiert wurde und auch sehr leistungsfähig ist, waren einige Einschränkungen ausschlaggebend, daß im Verlaufe der Diplomarbeit ein neuer Debugger entwickelt wurde. Die wesentlichen Gründe waren:

- zu frühe Ausführung von Befehlen
- keine Verfolgung von source-Befehlen
- keine Unterstützung von Autoload-Funktionen
- keine „Next-Statement“-Funktion
- keine bedingten Breakpoints
- kein Editieren des Skriptes, während des Debuggen möglich
- unzureichende Darstellung von Variablen, Stack, Janusobjekten
- DLL-Schnittstelle von Janus kann nicht genutzt werden
- nicht ausreichende Flexibilität und Anpaßbarkeit



File: i:\Speech_Projects\P1\S1\test\test.tcl

1

0

```
  ◇ regsub -all {\#[^\#]+\#} $corr "" corr
  ◇
  ◇ #    set ali [lindex [align $corr $hypo] 9]
  ◇ #jr
  ◇ set ali [lindex [lindex [rawAlign $corr $hypo] 1] end]
  ◇ #    set errN 0
  ◇ set errN [expr $ali/1000]
  ◇ #    foreach i $ali { if { $i != "c" } { incr errN } }
  ◇ return [expr 100.0 * (1.0 - $errN.0 / [llength $corr])]
◆ proc report {text corr hypo}
  ◇ global output
  ◇ set wa ""
  ◆ if { $output(WA) }
    ◇ set wa [recogRate $corr $hypo]
  ◇ puts "$text $wa"
  ◆ if { $output(HYPO) }
◇
◇ # ======================================================================
◇ #    Test -- here we go (mixed Detlef and TK)
◇ # ======================================================================
◆ if {![file exists $hypoPath]}
  ◇ exec mkdir $hypoPath
◇
◆ if { ! $output(INFO) }
◇
```

Abbildung 4.20: Outline-Debugger

Der in Abbildung 4.21 abgebildete Debugger ist in einen speziell dafür entwickelten hierarchischen Editor integriert, der zu einer vollständigen Entwicklungsumgebung ausgebaut werden kann. Obwohl die Entwicklung noch nicht abgeschlossen ist, sind die wesentlichen Merkmale schon vorhanden.

Der hierarchische Editor gibt eine gliederungsartige Zusammenfassung über den Programmcode, indem er zuerst nur die oberste Ebene des Skriptes anzeigt und die Details verbirgt. So kann man sich schnell einen Überblick verschaffen. Durch einen Doppelklick auf eine Zeile wird die nächste Stufe anzeigt. Solche Sichtweisen sind z.B. bei Dateimanagern (Norton Commander, Microsoft Explorer) und ähnlichen Programmen üblich. Ein hierarchischer Editor zwingt zu einer sauberen Gliederung. Der Editor baut beim Einlesen eines Tcl-Skript die hierarchische Ansicht anhand einer Sprachbeschreibung von Tcl auf. Beim Speichern wird eine leserliche, eingerückte Tcl-Datei aus den Hierarchieebenen erzeugt. Die Hierarchieebenen entsprechen in etwa dem intuitiven Einrücken beim Programmieren in Tcl. So bilden momentan die Schlüsselwörter proc, while, for, foreach, if, else, elseif, switch, uplevel, eval, source und catch eine neue, untergeordnete Ebene. Bis jetzt sind nur diese direkten Tcl-Konstrukte berücksichtigt, eine Erweiterung beispielsweise um das Janus-Konstrukt „doParallel“ aber ohne weiteres denkbar. Die Rauten am Zeilenanfang zeigen jeweils an, ob sich hinter der Zeile noch eine weitere aufklappbare Ebene verbirgt. Sie sind in diesem Fall schwarz ausgefüllt.

Während des Debuggens kann der Quellcode verändert oder korrigiert und der Debugger angewiesen werden, an einer beliebigen Stelle wieder aufzusetzen, um die letzten, korrigierten Schritte erneut zu debuggen. Dieses Vorgehen erlaubt einen sehr effektiven Programmtest.

Der Editor umfaßt dabei sehr komfortable Editiermöglichkeiten. Neben Cut & Paste können per Drag & Drop einzelne Zeilen einfach an eine andere Stelle verschoben werden, auch dann, wenn sie sich in unterschiedlichen Hierarchieebenen befinden. Es entfällt die lästige Nachbearbeitung des Ein-/Ausrückens. Neben den allgemein üblichen Standardfunktionen beherrscht der Editor folgende Tastatur-/Mauskommandos:

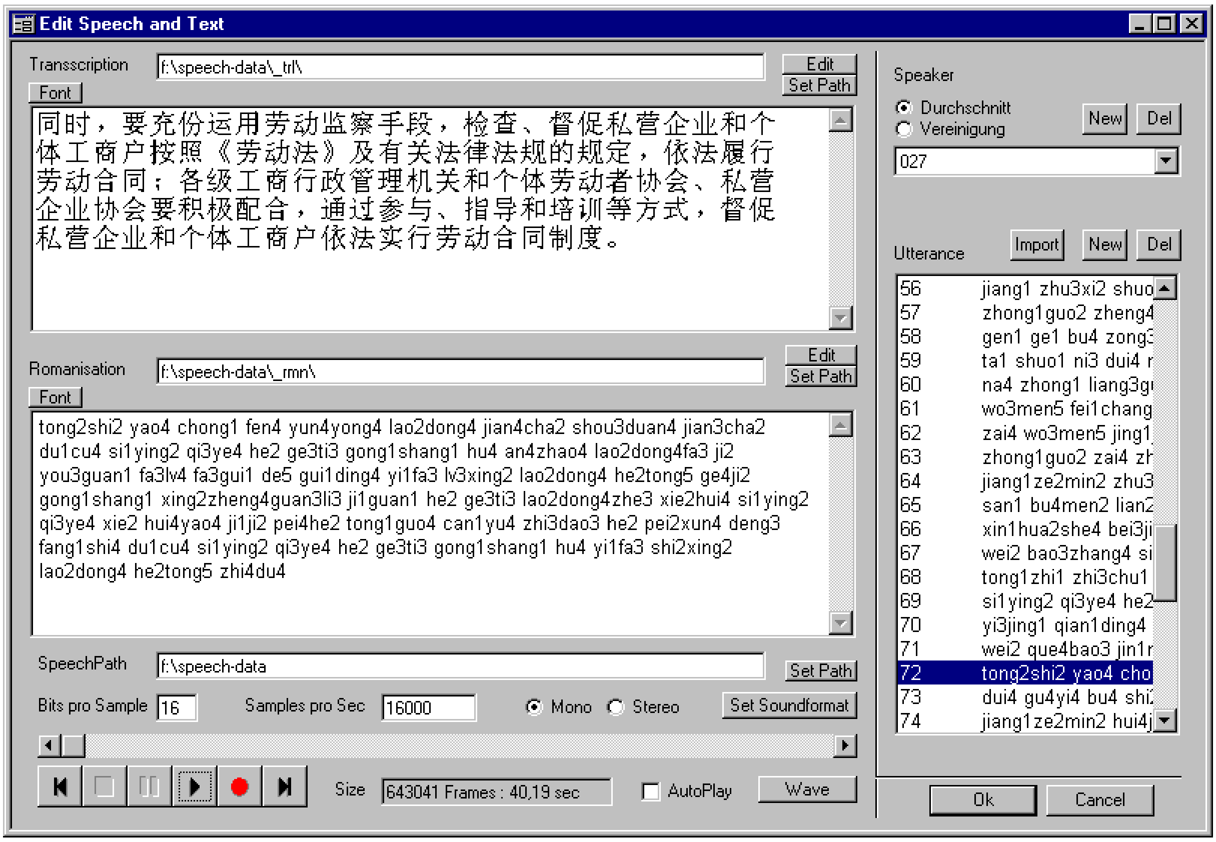
```
Einfachklick wechselt in den Editiermodus
Enter wechselt in den Selectmodus
Im Editiermodus:
   Shift + Enter erzeugt Neue Zeile
Im Selectmodus:
   Alt + ↑ Objekt um jeweils eine Zeile nach oben schieben
   Alt + ↓ Objekt um jeweils eine Zeile nach unten schieben
   Alt + ←Objekt um jeweils eine Hierarchieebene nach außen schieben
   Alt + →Objekt um jeweils eine Hierarchieebene nach innen schieben
   Einfg   Objekt einfügen
   Entf    Objekt löschen
   +       Eine Hierarchieebene aufklappen (Expand)  (=Doppelklick)
   -       Alle nachfolgenden Hierarchieebene zuklappen (Colapse)  (=Doppelklick)
   *       Alle nachfolgenden Hierarchieebene aufklappen (Expand All)
   /       Alle Hierarchieebene des Skripts zuklappen (Colapse All)
```

Eine Suchfunktion durchsucht zyklisch den Quellcode und klappt bei Bedarf automatisch Hierarchieebenen auf. Der integrierte Debugger markiert Befehle vor ihrer Ausführung und kann sowohl Source-Befehle, als auch per Autoload geladene Funktionen verfolgen, indem die entsprechenden Skripte geladen und nach Beendigung wieder entladen werden. Zu jedem Zeitpunkt kann die nächste auszuführende Zeile frei ausgewählt werden. Die Einzelschrittverarbeitung erlaubt anzugeben, ob der nächste Befehl als Block ausgeführt werden soll (Step Over) oder ob in ihn hinein debuggt werden soll (Step Into). Im Run-Mode wird das Skript bis zum nächsten Breakpoint oder bis zum Ende ausgeführt. Den Breakpoints können einfache

Bedingungen zugeordnet werden, die zum Anhalten erfüllt sein müssen. Über die DLL-Schnittstelle von Janus können alle Daten wie Variablen, Stack und Janus-Objekte zu jedem Zeitpunkt ausgelesen werden. Einer entsprechenden Repräsentation und beliebigen Integration in das Janus-UI steht somit nichts im Wege. Besonders angenehm ist auch die Möglichkeit vom Tcl-Debugger zum Visual C-Debugger nahtlos übergehen zu können, so daß Janusobjekte auch weiter in der C-Implementierung verfolgt werden können. Wenn dies nicht ausreicht kann weiter in die Tcl/Tk-Quellen und die Windows-API debuggt werden.
Der jetzige Entwicklungsstand hat noch den Status eines Prototyps, konnte aber besonders während der Windowsportierung entscheidende Hilfe leisten. Ein weiterer Ausbau in Richtung vollständige Entwicklungsumgebung (grafische Tk-Unterstützung ...) wäre denkbar.

### 4.3.6 Audio- und Transkriptionsbearbeitung

Um die für den Spracherkenner notwendigen Trainingsdaten zu erhalten, können mit einem hochwertigen Aufnahmegerät zum Beispiel vorgelesene Texte aufgezeichnet werden. Der geringste Qualitätsverlust läßt sich mit digitaler Aufzeichnung mittels Dat-Recorder erreichen, da dann die Daten digital auf ein Computersystem überspielt werden können. Die Samplingrate und Bitbreite sollte während der Aufzeichnung möglichst hoch sein, da ein Konvertieren nur in Richtung niedrigere Samplingrate und Bitbreite verlustfrei möglich ist. Aus dem gleichen Grund sollte auch die Aufnahme in Stereo erfolgen. So hat man immer die Möglichkeit Spracherkenner mit den unterschiedlichsten Audioparametern zu konstruieren. Nachdem die Aufnahmen auf ein Computersystem überspielt wurden, müssen sie in einzelne Sounddateien aufgesplittet werden, so daß für jede Äußerung eine eigene Datei mit aussagekräftigen Dateinamen entsteht. Da jedem Sprecher mehr oder weniger Fehler unterlaufen, müssen Korrekturen angewandt werden. Da Korrekturen am Text wesentlich einfacher als Korrekturen an den Audiofiles durchzuführen sind, wird der Text an die Audiodateien angepaßt. Dazu vergleicht ein Testhörer die Texte mit den Äußerungen. Um das Testhören und Korrigieren zu vereinfachen, wurde das unten dargestellte Werkzeug entwickelt, welches Text und Audio synchronisiert.



Transscription: f:\speech-data_trl\

同时，要充份运用劳动监察手段，检查、督促私营企业和个体工商户按照《劳动法》及有关法律法规的规定，依法履行劳动合同；各级工商行政管理机关和个体劳动者协会、私营企业协会要积极配合，通过参与、指导和培训等方式，督促私营企业和个体工商户依法实行劳动合同制度。

Romanisation: f:\speech-data_rmn\

tong2shi2 yao4 chong1 fen4 yun4yong4 lao2dong4 jian4cha2 shou3duan4 jian3cha2 du1cu4 si1ying2 qi3ye4 he2 ge3ti3 gong1shang1 hu4 an4zhao4 lao2dong4fa3 ji2 you3guan1 fa3lv4 fa3gui1 de5 gui1ding4 yi1fa3 lv3xing2 lao2dong4 he2tong5 ge4ji2 gong1shang1 xing2zheng4guan3li3 ji1guan1 he2 ge3ti3 lao2dong4zhe3 xie2hui4 si1ying2 qi3ye4 xie2 hui4yao4 ji1ji2 pei4he2 tong1guo4 can1yu4 zhi3dao3 he2 pei2xun4 deng3 fang1shi4 du1cu4 si1ying2 qi3ye4 he2 ge3ti3 gong1shang1 hu4 yi1fa3 shi2xing2 lao2dong4 he2tong5 zhi4du4

SpeechPath: f:\speech-data

Bits pro Sample: 16

Samples pro Sec: 16000

● Mono ○ Stereo

Size: 643041 Frames : 40,19 sec

☐ AutoPlay

Speaker

● Durchschnitt
○ Vereinigung

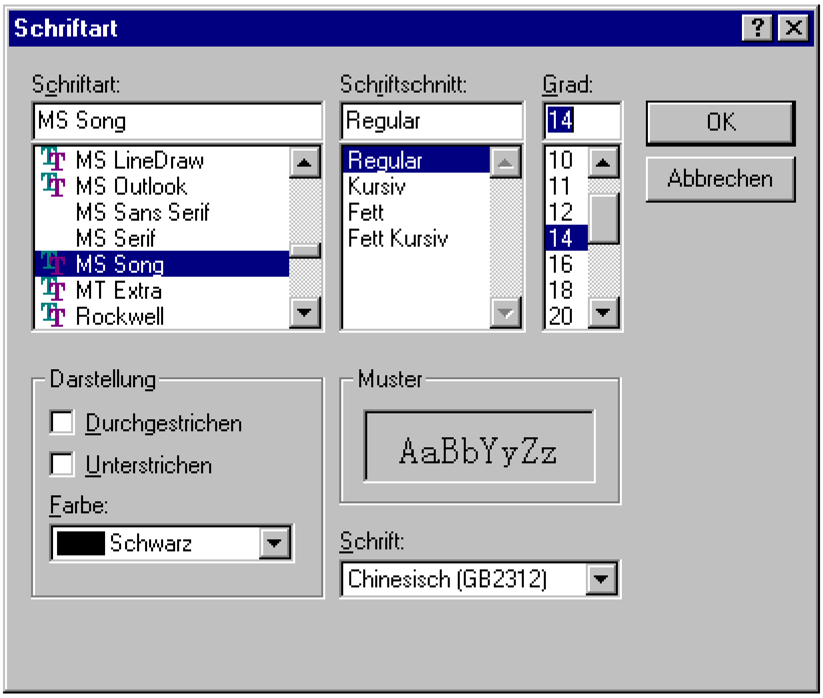
027

Utterance

| | |
|---|---|
| 56 | jiang1 zhu3xi2 shuo |
| 57 | zhong1guo2 zheng4 |
| 58 | gen1 ge1 bu4 zong |
| 59 | ta1 shuo1 ni3 dui4 |
| 60 | na4 zhong1 liang3g |
| 61 | wo3men5 fei1chang |
| 62 | zai4 wo3men5 jing1 |
| 63 | zhong1guo2 zai4 zh |
| 64 | jiang1ze2min2 zhu3 |
| 65 | san1 bu4men2 lian |
| 66 | xin1hua2she4 bei3ji |
| 67 | wei2 bao3zhang4 si |
| 68 | tong1zhi1 zhi3chu1 |
| 69 | si1ying2 qi3ye4 he2 |
| 70 | yi3jing1 qian1ding4 |
| 71 | wei2 que4bao3 jin1r |
| 72 | tong2shi2 yao4 cho |
| 73 | dui4 gu4yi4 bu4 shi |
| 74 | jiang1ze2min2 hui4j |

Abbildung 4.21: Play and Record - Tool

Im rechten Fensterteil (Abb. 4.21) kann mit der Combobox „Sprecher” einer der verfügbaren Sprecher ausgewählt werden. „Durchschnitt“ zeigt nur Sprecher an, die sowohl Audio- als auch Textdaten besitzen. In der darunterliegenden Listbox werden dann alle Äußerungen dieses Sprechers mit Nummer und Textanfang aufgelistet. Nach Auswahl einer Äußerung erscheinen dann der Orginaltext, sowie eine Lautschriftumsetzung in zwei Fenstern. Durch Betätigen der Audiotasten kann, ähnlich eines Kassettenrecorders, die Äußerung wiedergegeben oder an eine bestimmte Stelle „gespult“ werden. Die sich darüber befindende Scrollbar zeigt während der Aufnahme die relative Position im Audiofile an. Sie kann im Pausemodus an eine andere Position gezogen werden, um dort mit der Wiedergabe fortzusetzen. Ein Klick/Tastendruck auf „Bildhoch/Bildrunter“ entspricht dabei 1 Sekunde während ein Klick auf „Zeilehoch/Zeilerunter“ 0,1 Sekunden entspricht. Im Size-Feld werden die Anzahl der Frames und die Dauer in Sekunden angezeigt. Durch Doppelklick auf eine Äußerung in der Listbox wird sofort mit der Wiedergabe begonnen. Bei Auswählen von „Auto Play“ wird nach Beenden der Wiedergabe automatisch die nächste Äußerung geladen und wiedergegeben. Als Audioformate werden das PCM- und WAV-Format, bei automatischer Erkennung, unterstützt. Die Bittiefe, Samplingrate bzw. Mono/Stereo können in weiten Schranken eingestellt werden.



Schriftart: MS Song
Schriftschnitt: Regular
Grad: 14
☐ Durchgestrichen
☐ Unterstrichen
Farbe: Schwarz
Schrift: Chinesisch (GB2312)

Schriftart
MS LineDraw
MS Outlook
MS Sans Serif
MS Serif
MS Song
MT Extra
Rockwell
Regular
Kursiv
Fett
Fett Kursiv
10
11
12
14
16
18
20
OK
Abbrechen
Darstellung
Muster
AaBbYyZz

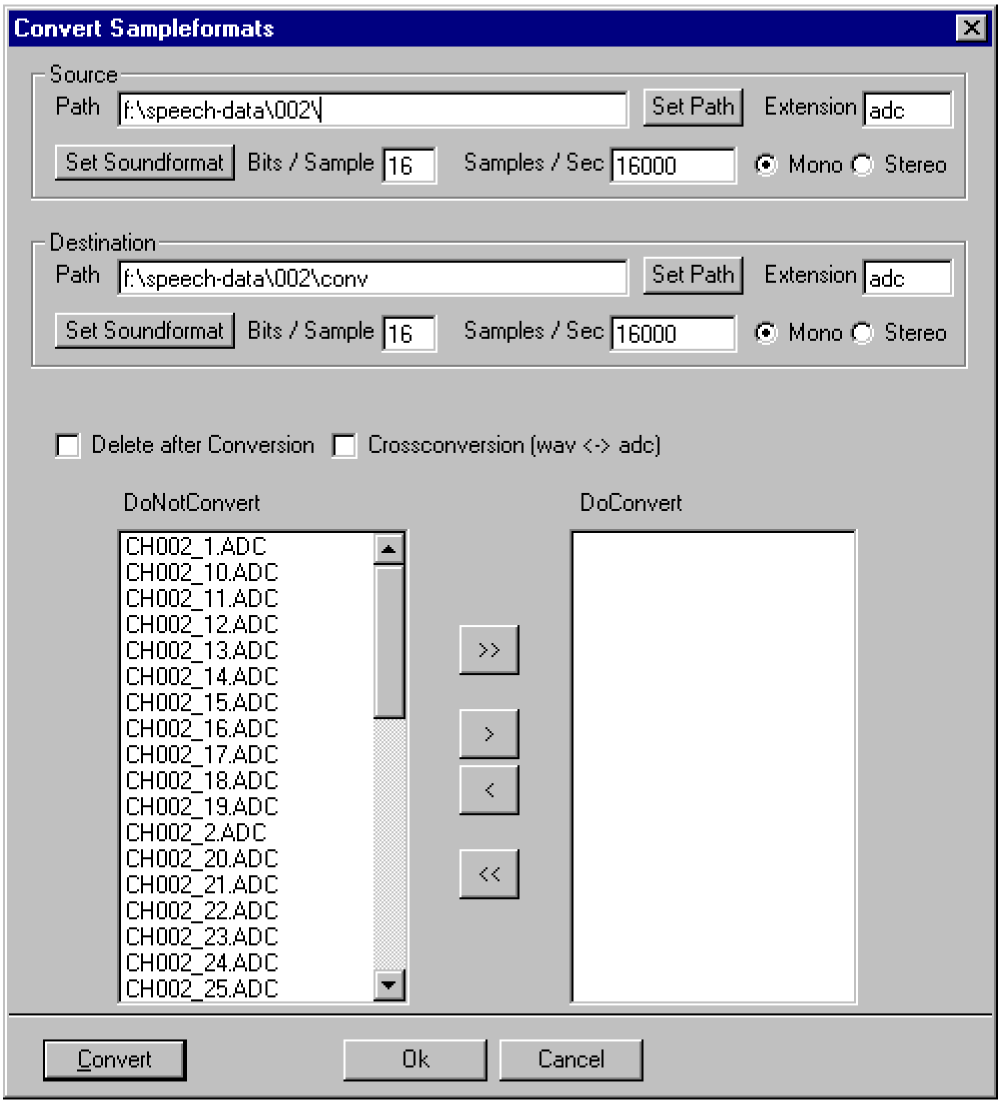
Abbildung 4.22: Fontauswahl

In den Textfeldern können Aussprachefehler, Versprecher, Räuspern, Wiederholungen und dergleichen korrigiert werden. Um universell einsetzbar zu sein, wurde für die Textfelder eine Fontauswahl vorgesehen (Abb. 4.22), so daß beliebige Zeichensätze darstellbar sind (in der Abbildung 4.21 sind dies chinesische Kurzzeichen in der GuoBiao-Kodierung, darunter die zugehörige Pinyin-Umsetzung).

Neben der Korrekturunterstützung erlaubt das Werkzeug auch direkte Sprachaufnahmen synchron zum angezeigten Text. Somit entfällt das zeitaufwendige Überspielen der Aufnahmen auf ein Computersystem und die dortige Zerlegung in Einzeldateien. Die Verwendung des Systems auf einem Notebook erlaubt dabei eine ähnliche Mobilität, wie die Verwendung eines Dat-Rekorders. Die Sprecherbuttons „New“ und „Del“ legen einen neuen Sprecher an oder löschen einen vorhandenen, während Äußerungsbuttons „New“ und „Del“ Äußerungen anlegen bzw. löschen.
Der „Wave“-Button lädt das aktuelle Audiofile in einen Audioeditor, um es dort bearbeiten zu können. Neben dem Wegschneiden von Abschnitten können so auch Rauschfilter oder andere Filter einfach angewendet werden.

### 4.3.7 Audiokonvertierung

Wie im letzten Unterkapitel schon erwähnt, ist es von Vorteil die Audioaufnahmen immer mit maximaler Qualität aufzuzeichnen, um Spracherkenner mit unterschiedlichen Audioparametern trainieren zu können. Um nun aus den Orginalaudiodaten Audiodaten eines speziellen Formates zu erzeugen, müssen diese konvertiert werden.



Convert Sampleformats

Source
Path f:\speech-data\002\
Extension adc
Bits / Sample 16
Samples / Sec 16000
● Mono ○ Stereo

Destination
Path f:\speech-data\002\conv
Extension adc
Bits / Sample 16
Samples / Sec 16000
● Mono ○ Stereo

☐ Delete after Conversion ☐ Crossconversion (wav <-> adc)

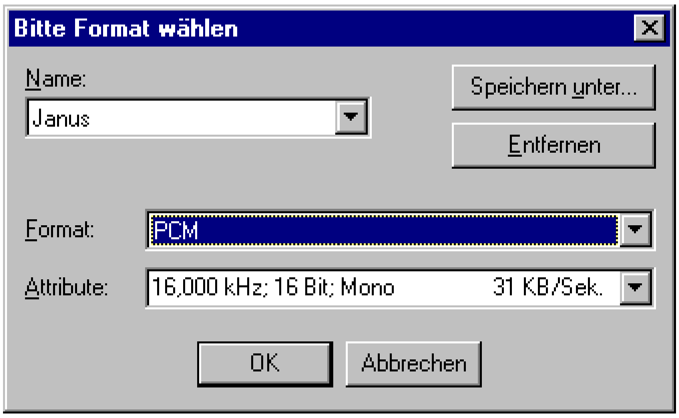
DoNotConvert
CH002_1.ADC
CH002_10.ADC
CH002_11.ADC
CH002_12.ADC
CH002_13.ADC
CH002_14.ADC
CH002_15.ADC
CH002_16.ADC
CH002_17.ADC
CH002_18.ADC
CH002_19.ADC
CH002_2.ADC
CH002_20.ADC
CH002_21.ADC
CH002_22.ADC
CH002_23.ADC
CH002_24.ADC
CH002_25.ADC

DoConvert

Abbildung 4.23: Audioformat-Konverter

Nun erlauben zwar viele kommerzielle Audioprogramme eine Formatkonvertierung. Aber es muß jede einzelne Datei geladen, konvertiert und wieder gespeichert werden. Auch Geschwindigkeit und Konvertierungsqualität lassen oft zu wünschen übrig. Neben diesen Überlegungen spielt natürlich neben dem Anschaffungspreis auch die Flexibilität und Integrationsfähigkeit einer eigenen Lösung die entscheidende Rolle. Somit wurde ein neuer Audioformat-Konverter, basierend auf Microsofts Multimedia-Schnittstelle, entworfen.

Diesem Audiokonverter werden Quell- und Zielverzeichnis, sowie Dateiendung der Audiofiles mitgeteilt. Dann wird das Audioformat mit Bittiefe, Samplingrate und Stereo/Mono jeweils für Quelle und Ziel angeben. Über die vorhandenen Listboxen können anschließend die zu konvertierenden Dateien ausgewählt und durch den „Convert“-Button die Batch-Konvertierung begonnen werden. Dabei erreicht man auf einem Standardsystem beim Downsampeln ca. 40 fache Echtzeit bei guter Qualität.
Die Konvertierung, wie auch die Audiowiedergabe setzen auf Microsofts Low Level Audio Programmierschnittstelle auf. Einige Implementierungsdetails werden im nächsten Unterkapitel beschrieben.



Bitte Format wählen

Name: Janus

Format: PCM

Attribute: 16,000 kHz; 16 Bit; Mono 31 KB/Sek.

Abbildung 4.24: Audioformat wählen

Über die Checkbox „Crosskonversion“ können PCM-Dateien in WAV-Dateien und umgekehrt gewandelt werden. Der Button „Set Soundformat“ öffnet den oben abgebildeten Dialog (Abb. 4.24), in dem Audioparameter unter einem Namen zusammengefaßt werden können. Hier werden z.B. unter dem Namen „Janus“ die Audioparameter PCM mit Samplingrate 16 kHz, Bittiefe 16 Bit, Modus Mono zusammengefaßt. Somit kann schnell zwischen definierten Audioformaten gewechselt werden. Diese Festlegungen gelten übrigens systemweit, so daß andere Audioprogramme diese Festlegungen auch benutzen können. Auch können über diesem Dialog Kompressionsverfahren und andere Filter angewendet werden. So sind unter anderem folgende teils kommerzielle Formate in jeweils mehreren Ausprägungen möglich:

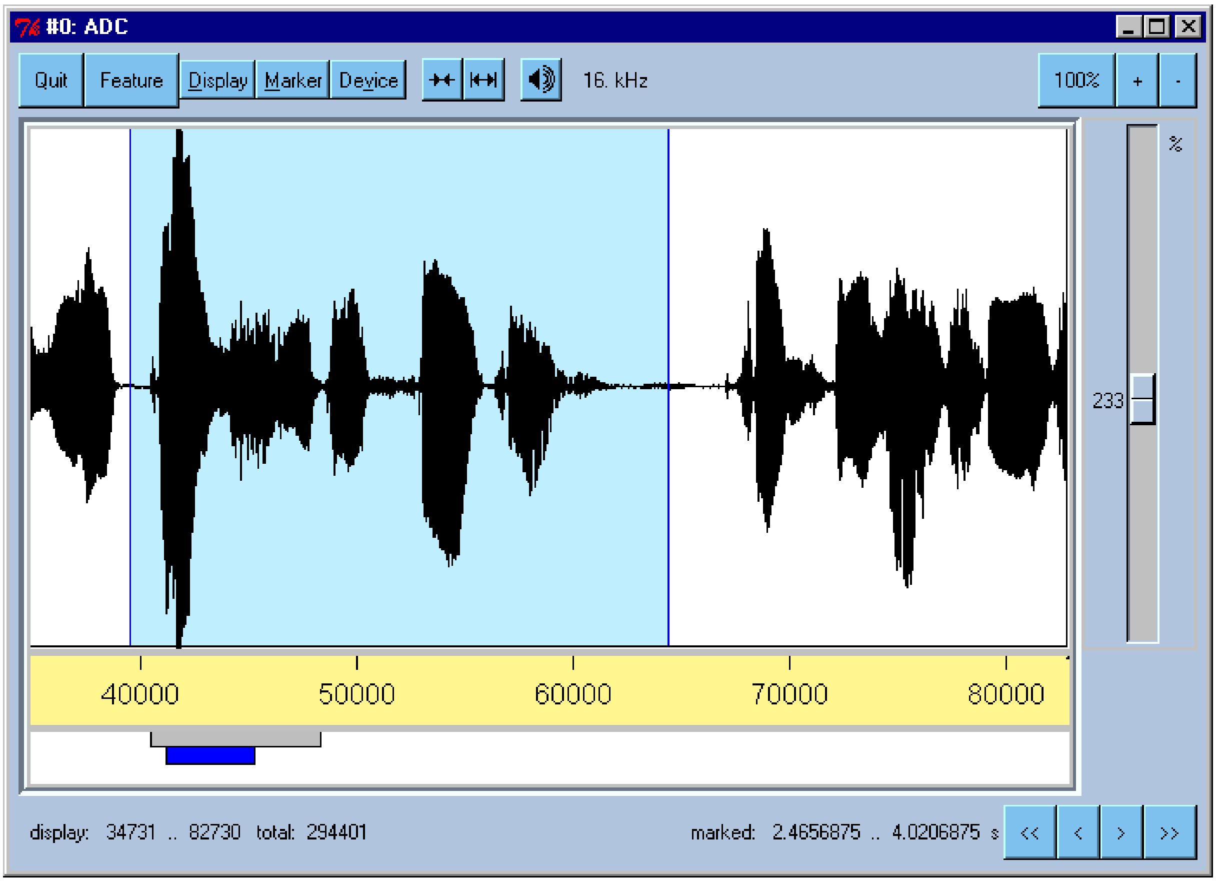
- CCITT A-Law
- CCITT u-Law
- DSP Group TrueSpeech ™
- GSM 6.10
- IMA ADPCM
- Lernout & Hauspie CELP
- Lernout & Hauspie SBC
- Microsoft ADPCM
- Microsoft G.723.1
- MPEG Layer-3
- MSN Audio
- PCM
- VivoActive G.723.1
- VivoActive Siren
- Voxware MetaSound
- Voxware MetaVoice

### 4.3.8 Die Audioschnittstelle

Für die Windowsversion von Janus war es nötig, Sounddateien abspielen zu können. Unter Unix existiert dafür ein Programm, dem die Sounddatei und einige Audioparameter übergeben werden. So wurde auch unter Windows ein kleines Tool für diese Aufgabe entwickelt. „Play.exe“ kann PCM- und WAV-Files wiedergeben. Zusätzlich wurde ein Play-Befehl in Janus/Tcl integriert. Eine Eingabe von `Play` oder `Play -help` zeigt die erforderlichen Parameter an.

```
play [-r Samplingrate] [-c Channels] [-b Bits] [-s Startpos] [-e Endpos]
     [-help] -f Feature|-d Datei  (autom. Wave/ADC-Erkennung)
```

Wenn keine Samplingrate, Channels, Bits, Startpos und Endpos angegeben werden, werden die vordefinierten Standardwerte benutzt. Dies sind 16 kHz Samplingrate, Mono, 16 Bit, von der Startposition 0 bis zum Ende. Die Position bezieht sich auf die Frameanzahl (im Standardfall 16 Bit-Worte). Mit Positionsangaben können Soundfiles auch ausschnittsweise angehört werden und durch Angabe eines Features aus einem FeatureSet kann dieses ohne Umweg über die Festplatte sofort ausgegeben werden, so daß keinerlei zeitliche Verzögerung entsteht. Davon macht z.B. ein angepaßter Wave/Feature-Betrachter aus den Janus-Tools Gebrauch, der Soundausschnitte direkt anhand des zugrundeliegenden Features wiedergeben kann.



display: 34731 .. 82730 total: 294401

marked: 2.4656875 .. 4.0206875 s

Abbildung 4.25: Wave/Feature-Betrachter

Ein weiterer Schritt in der Entwicklung einer Audioschnittstelle für Janus war die Integration einer Aufnahmefunktion. Diese verursachte unerwartete Probleme mit dem MCI-Interface, als deren Ursache sich das Zusammenspiel der Soundblaster Treiber (sowohl bei Soundblaster 16, 16PnP, AWE32 und AWE64) mit Windows NT beim Zugriff mittels High Level Audiobefehlen, herausstellte. In den Newsgroups waren keine Lösungen bekannt, und der Support von Creative konnte auch nicht weiterhelfen. Somit mußte auf das Low Level Audiointerface zurückgegriffen werden, was einen Neuentwurf der Audioschnittstelle erforderte. Als positive Auswirkungen entstanden eine flexible DLL-Schnittstelle, die Möglichkeit im Hintergrund aufzunehmen und wiederzugeben, Aufnahme/Wiedergabe direkt von/auf Festplatte, keine Längenbegrenzung, PCM/WAV-Wandlung, sowie Audiokonvertierungen durchzuführen.

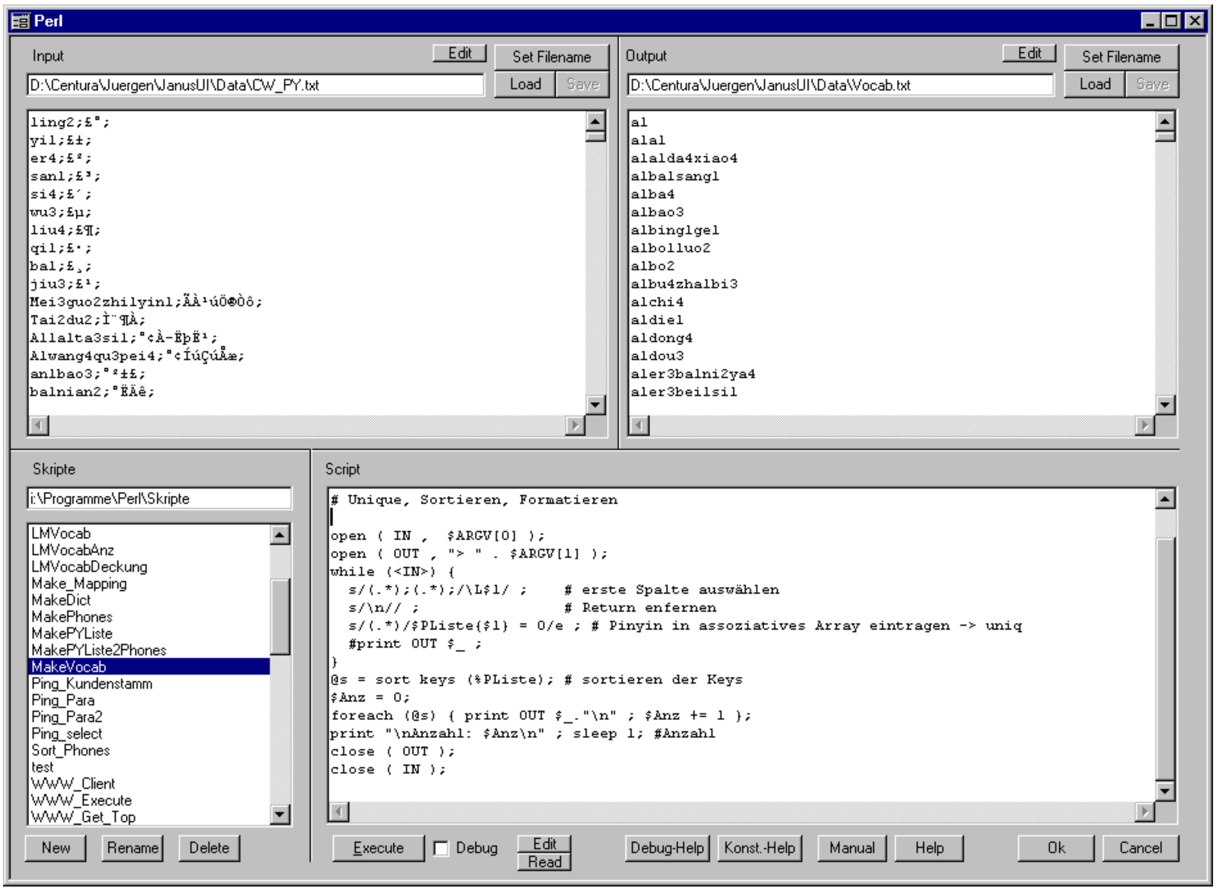
Die DLL-Schnittstelle ist sehr einfach aus jedem beliebigen Windowsprogramm heraus zu verwenden. In der Tabelle unten sind die fünf exportierten Befehle angegeben.

| **Befehl** | **Beschreibung** |
|---|---|
| `Stop()` | Beendet Aufnahme oder Wiedergabe |
| `Record ( sDateiname, bWAV/PCM )` | Startet Aufnahme in Datei „Dateiname“<br>Dateiformat WAV oder PCM |
| `Play ( sDateiname, nStartpos, nEndpos )` | Startet Wiedergabe aus Datei „Dateiname“<br>Von Position Startpos bis Endpos<br>(Endpos –1 entspr. Dateiende) |
| `SetWaveFormatDlg ( bOutConvert, bDlg, nCh, nSampleRate, nBitsPerSample )` | Setzt Quell- oder Zielformat<br>bOutConvert: Auswahl Ziel oder Quelle<br>bDlg: Flag für erweiterten Formatdialog<br>nCh: Kanalanzahl (1=Mono, 2=Stereo)<br>nSampleRate: Abtastrate<br>nBitsPerSample: Bittiefe |
| `Convert (sDateiIn, sDateiOut, nStartPos, nEndPos);` | Konvertiert Datei „DateiIn“ nach „DateiOut“ unter Berücksichtigung der Formatangaben. StartPos/EndPos erlauben nur einen Ausschnitt zu konvertieren. |

Alle Funktionen geben bei Problemen Fehlernummern zurück. Implementiert sind die Funktionen als Methoden der Klasse Recorder, um die Wiederverwendbarkeit zu vereinfachen. Beim Erzeugen eines Recorderobjektes wird ein Hintergrundprozeß (Server) gestartet, dieser wartet in einer Endlosschleife auf Ereignisse, die ihm durch einen Signalisierungsmechanismus mitgeteilt werden können. Beim Auftreten von Ereignissen ist es wichtig, daß ein gleichzeitiger Zugriff auf gemeinsame Objekte von Client und Server nur unter gegenseitigem Ausschluß stattfindet. Dies kann beispielsweise durch die Definition von kritischen Abschnitten, die nur ein Prozeß betreten darf, geschehen. Desweiteren muß die Synchronisation eines Ringpuffers gesteuert erfolgen. In einen Bereich des Ringpuffers schreibt beim Abspielen der Hauptprozeß (Client) Daten, während der Hintergrundprozeß Daten aus einem anderen Pufferbereich ausließt und zum Audiogerät weiterleitet, bzw. bei der Aufnahme in umgekehrter Richtung. Die Audiogeräteauswahl und Audioverarbeitung basiert auf den „Waveform-Functions“ und die Konvertierung auf den „Audio Compression Functions“ der Microsoft Multimedia API.

### 4.3.9 Die Perl-Umgebung

Zur Erzeugung und Aufbereitung von Daten, die zum Training eines Spracherkenners gebraucht werden, sind aufwendige „Textprocessing“-Verfahren notwendig. Keine Sprache hat in diesem Bereich mehr Verbreitung erfahren, als die von Larry Wall entwickelte Sprache Perl. Ähnlich wie bei Tcl/Tk sind hier die „Executables“ wie auch die „Sources“ frei erhältlich. Perl wurde auf annähernd jede Plattform portiert und ist dabei, viele Aufgaben, die bisher in Shellsprachen realisiert wurden, portabel und meist mit wesentlich höherer Performance abzulösen. Perl hat sich nicht nur zu einer Obermenge von Shellvarianten mit ihren zugehörigen Unix-Tools gemausert, sondern wird auch als Programmiersprache eingesetzt. Besondere Verbreitung hat Perl in Verbindung mit dem CGI (Common Gateway Interface) zur dynamischen Erzeugung von HTML-Seiten erfahren.
Nachdem erste Vorverarbeitungen des Sprachkorpus unter Tcl stattfanden, wurde später aufgrund von Performancegewinn und eleganterer Darstellung fast ausschließlich Perl eingesetzt. Um Perl besser in die Janus-Benutzeroberfläche zu integrieren, wurde eine Perlumgebung entwickelt, die neben der Verwaltung der Skripte auch erlauben sollte, visuell die Änderungen in den Textdateien verfolgen zu können.



Input: D:\Centura\Juergen\JanusUI\Data\CW_PY.txt

```
ling2;£°;
yi1;£±;
er4;£²;
san1;£³;
si4;£´;
wu3;£µ;
liu4;£¶;
qi1;£·;
ba1;£¸;
jiu3;£¹;
Mei3guo2zhi1yin1;ÃÀ¹úÖ®Òô;
Tai2du2;Ì¨¶À;
A1la1ta3si1;°¢ÀËþË¹;
A1wang4qu3pei4;°¢ÍúÇúÅæ;
an1bao3;°²±£;
ba1nian2;°ËÄê;
```

Output: D:\Centura\Juergen\JanusUI\Data\Vocab.txt

```
a1
a1a1
a1a1da4xiao4
a1ba1sang1
a1ba4
a1bao3
a1bing1ge1
a1bo1luo2
a1bo2
a1bu4zha1bi3
a1chi4
a1die1
a1dong4
a1dou3
a1er3ba1ni2ya4
a1er3bei1si1
```

Skripte: i:\Programme\Perl\Skripte

- LMVocab
- LMVocabAnz
- LMVocabDeckung
- Make_Mapping
- MakeDict
- MakePhones
- MakePYListe
- MakePYListe2Phones
- MakeVocab
- Ping_Kundenstamm
- Ping_Para
- Ping_Para2
- Ping_select
- Sort_Phones
- test
- WWW_Client
- WWW_Execute
- WWW_Get_Top

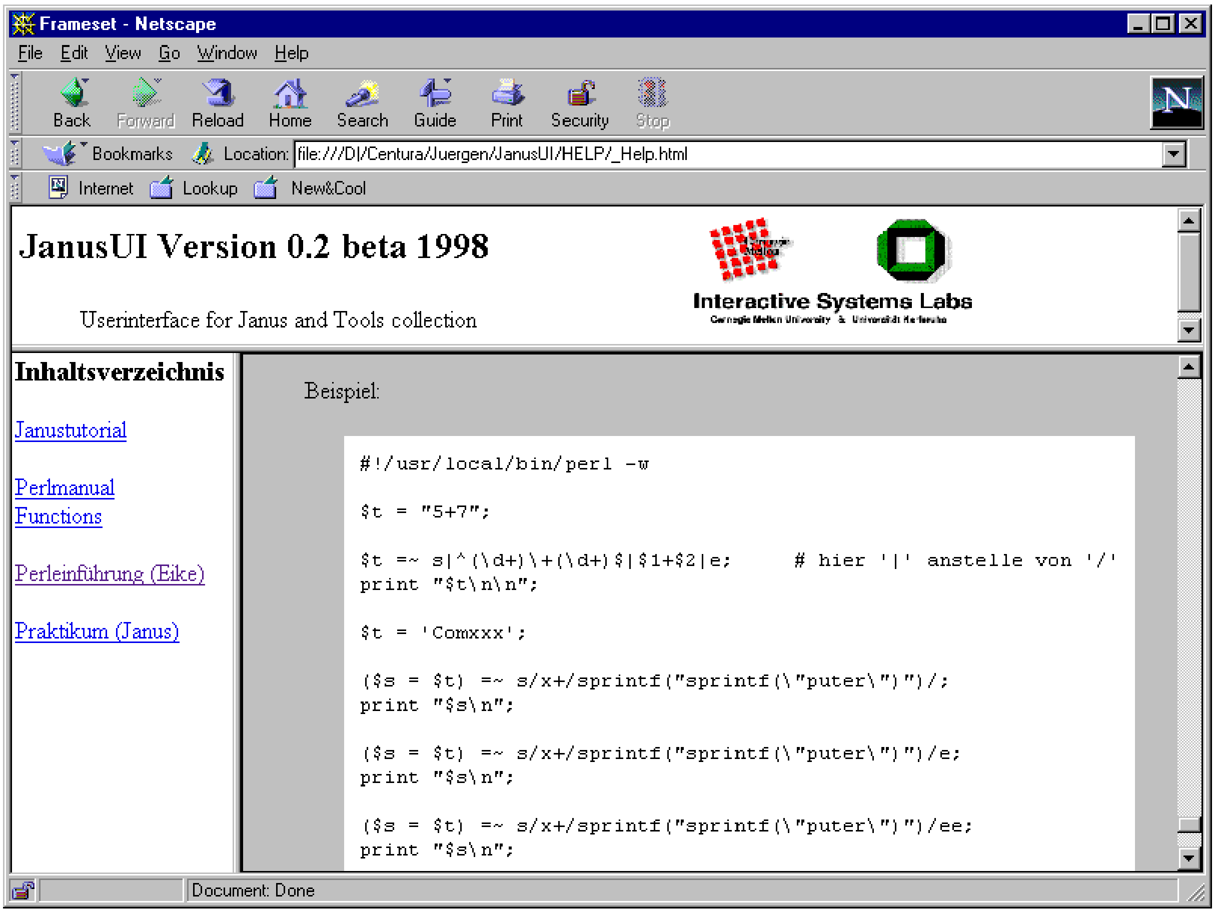
```
# Unique, Sortieren, Formatieren

open ( IN ,  $ARGV[0] );
open ( OUT , "> " . $ARGV[1] );
while (<IN>) {
  s/(.*);(.*);/\L$1/ ;    # erste Spalte auswählen
  s/\n// ;                # Return enfernen
  s/(.*)/$PListe{$1} = 0/e ; # Pinyin in assoziatives Array eintragen -> uniq
  #print OUT $_ ;
}
@s = sort keys (%PListe); # sortieren der Keys
$Anz = 0;
foreach (@s) { print OUT $_."\n" ; $Anz += 1 };
print "\nAnzahl: $Anz\n" ; sleep 1; #Anzahl
close ( OUT );
close ( IN );
```

Abbildung 4.26: Perlumgebung

Das obere linke Fenster enthält dazu die zu verarbeitenden Daten, die das in der Listbox ausgewählte Skript verarbeitet und die Ausgabe dann im oberen rechten Fenster darstellt. Änderungen am Skript können im Quelltextfenster erfolgen, die dann sofort berücksichtigt werden. Dies gewährleistet einen schnellen Code-Test-Zyklus. Perldebugger, wie auch Hilfetexte sind einfach zugänglich. Obwohl die meisten Perlskripte nicht sprachabhängig sind, wird auf sie erst im nächsten Kapitel beim Aufbau eines chin. Erkenners eingegangen.

### 4.3.10 Hilfesystem

Um effektiv mit einem Werkzeug gleich welcher Art zu arbeiten, nimmt mit zunehmender Komplexität des Werkzeuges die Wichtigkeit der Dokumentation überproportional zu. So sollte auch dieser Aspekt berücksichtigt werden und alle verfügbare Hilfetexte über ein zentrales Inhaltsverzeichnis übersichtlich zugänglich gemacht werden. Als Basis dafür wurde HTML gewählt. Die Startseite kann über den Hilfebutton in JanusUI erreicht werden. (Das Hilfe-system ist nur ein erster Ansatz, das inhaltlich noch stark erweitert und um Suchfunktionen und Kontextsensivität ergänzt werden sollte.)


Frameset - Netscape
File
Edit
View
Go
Window
Help
Back
Forward
Reload
Home
Search
Guide
Print
Security
Stop
Bookmarks
Location:
Internet
Lookup
New&Cool
Interactive Systems Labs
Carnegie Mellon University & Universität Karlsruhe
Document: Done

Location: file:///D|/Centura/Juergen/JanusUI/HELP/_Help.html

**JanusUI Version 0.2 beta 1998**

Userinterface for Janus and Tools collection

**Inhaltsverzeichnis**

Janustutorial

Perlmanual
Functions

Perleinführung (Eike)

Praktikum (Janus)

Beispiel:

```
#!/usr/local/bin/perl -w

$t = "5+7";

$t =~ s|^(\d+)\+(\d+)$|$1+$2|e;     # hier '|' anstelle von '/'
print "$t\n\n";

$t = 'Comxxx';

($s = $t) =~ s/x+/sprintf("sprintf(\"puter\")")/;
print "$s\n";

($s = $t) =~ s/x+/sprintf("sprintf(\"puter\")")/e;
print "$s\n";

($s = $t) =~ s/x+/sprintf("sprintf(\"puter\")")/ee;
print "$s\n";
```

Abbildung 4.27: Startseite des Hilfesystems

# 5 Chinesische Spracherkennung

In den letzten drei Kapiteln wurden die Grundlagen zum Aufbau eines chinesischen Spracherkenners mittels des Janussystems gelegt. In diesem Kapitel sollen darauf aufbauend verschiedene Ansätze zur automatischen Spracherkennung diskutiert werden, von denen dann zwei näher untersucht werden.

Der chinesische Spracherkenner soll auf das, im vorherigen Kapitel vorgestellte, schon in vielen Sprachen erprobte Janussystem aufsetzen. Ein Ziel war somit die vorhandenen Ressourcen optimal auszunutzen, unter Beachtung der besonderen Merkmale der chinesischen Sprache.

Die Untersuchungen und der daran anschließende Aufbau verschiedener chinesischer Spracherkenner wurde unter Windows durchgeführt. Dabei wurde die im vorherigen Kapitel vorgestellte grafische Benutzeroberfläche mit ihren integrierten Werkzeugen verwendet.

## 5.1 Überblick über Ansätze chinesischer Spracherkennung

Da Chinesisch, anders als westliche Sprachen, keine wohldefinierten Wortgrenzen kennt, sondern alle Zeichen in einem Satz direkt aneinander reiht, ist es nicht offensichtlich, wie eine Bedeutungseinheit definiert werden soll. So kann eine Satzsegmentation z.B. anhand einzelner chinesischer Zeichen, speziell zu definierenden Wörtern, prosodischer Information [29] oder grammatischer Funktionalworte [16] geschehen. Für die vorliegende Arbeit wurde die Segmentation anhand von Wörtern vorgenommen. Die Länge der entstehenden Segmente liegt zwischen 1-10 Silben und stellt einen guten Kompromiß zwischen einer zu großen Anzahl von möglichen Segmenten und einem zu kleinen Kontext dar. Sie kommt damit auch der Satzsegmentation in westlichen Sprachen nahe. Die in einem Englisch-Chinesischen Wörterbuch definierte Abbildung von einem englischen Wort auf eine Folge von chinesischen Zeichen ist eine mögliche Definition von Worten. Nun ändern sich diese Definitionen evtl. schon von einem Wörterbuch zum nächsten. Besonders stark ist diese Änderung, wenn man von Englisch auf eine andere Sprache wechselt. Die Anzahl der Wortdefinitionen kann somit in weiten Grenzen variieren. Wichtig ist nur eine einmal festgelegte Definition immer konsistent anzuwenden. Für die akustische Repräsentation wird für jedes chinesische Zeichen eines Wortes eine Aussprachesilbe in der Pinyinumschrift verwendet. Chinesische Zeichen selbst sind ungeeignet, da einerseits viele Zeichen mehrere Aussprachen haben und umgekehrt eine Aussprachesilbe viele Zeichen bedeuten kann. D.h. die Abbildung zwischen chinesischen Zeichen und Pinyinumschrift muß funktional erfaßt werden. Bei der Wandlung von chinesischen Zeichen in die Pinyinumschrift geht Information verloren, da im Durchschnitt ca. acht Zeichen auf eine Pinyinaussprachesilbe abgebildet werden. Denn bei mehr als 10000 gebräuchlichen Zeichen gibt es nur ca. 400 Aussprachesilben, die mittels 5 Tönen zu ca. 1300 Aussprachesilben weiter differenziert werden. Die akustische Information reicht einem chinesischen Hörer, um das Gehörte niederschreiben zu können, wenn er erlerntes Kontext- und Metawissen einsetzen kann.

Beim konventionellen Ansatz zur Erfassung der Tonalität werden die Silbeninformation und die Toninformation getrennt analysiert und erst später wieder zusammengeführt [65][29][1]. Neuere Ansätze gehen allerdings dazu über, diese tonale Informationen zu integrieren [7][74], um die sonst in der Erkennungsphase erforderliche aufwendige Synchronisation von Silben- und Toninformation zu umgehen. Dabei kann der Pitch, der als aussagekräftiges Merkmal der

Tonalität angesehen wird, entweder als Teil des Merkmalvektors betrachtet werden, oder man verzichtet auf die Extraktion der Pitchinformation ganz und modelliert die Tonalität mittels eines Multistate-HMM, wie in dieser Arbeit.

Die Art und Weise sowie die Anzahl der verwendeten Phoneme variiert von 33 Phonemen [29] bis über 100 Phoneme [74]. Wesentliche Unterschiede existieren in der Definition der Phoneme. So wird der Ton unterschiedlich auf Phoneme einer Aussprachesilbe verteilt [79], oder eine Aussprachesilbe wird grundsätzlich in „initial“ und „final“ unterteilt [30]. In der Arbeit [30] wurde auch der Einfluß von „Intra-Syllable-Models“ untersucht und gezeigt, daß der Einfluß vom rechten Kontext wesentlich größer ist, als der vom linken.

Alle mir bekannten neueren Ansätze für die Erkennung größerer Vokabulare im Chinesischen sind HMM-basiert und haben 2-5 Zustände mit Gaußverteilungen [29][30][7][17][66][65][12] [1]. Zur Merkmalsdarstellung wird, soweit noch nicht geschehen, von LPC (linear predictive coding) nach MFCC (mel-scale cepstral coefficient) gewechselt, da Untersuchungen bis zu 17% weniger Fehler ergaben [12][30].

Die meisten Systeme wurden als Diktierlösung für ein großes Vokabular [65][29][12][74] oder als Abfragesystem für Informationssysteme [64][66] positioniert, da die Problematik der Dateneingabe bei zeichenbasierten Sprachen eine größere Rolle spielt.

## 5.2 Die Datensammlung

Im Rahmen des “GlobalPhone”-Projekts [53], welches sich mit dem Sammeln von Sprachdaten zur computerbasierten Sprachverarbeitung befaßt, wurden neben Arabisch, Japanisch, Koreanisch, Kroatisch, Portugiesisch, Russisch, Spanisch und Türkisch auch chinesische Sprachdaten gesammelt. Neben der chinesischen Hochsprache (Mandarin) mit der sich diese Arbeit beschäftigt, wurden auch Daten im Wu-Dialakt (Shanghai) gesammelt. Der Wu-Dialekt ist mit ca. 80 Mio. Sprechern, neben dem kantonesichen Dialekt und der Hochsprache, die größte Sprachgruppe.

Zur Sprachdatensammlung bieten sich Tageszeitungen als Datenquelle an, da sie ein breites Spektrum von Themen abdecken und oft online zugänglich sind, so daß ein Abtippen erspart wird. Für die chinesische Datensammlung wurde die chinesische Tageszeitung People‘s Daily（人民日报）, die unter http://www.snweb.com/pdhome.html online zugänglich ist, ausgewählt. Sie ist das offizielle Organ der Partei und im ganzen Land erhältlich. Aus ihr wurden politische Texte und Texte über Wirtschaftsthemen ausgewählt. Die HTML-Texte aus dem Zeitraum April bis Juni 96 wurden in reine Zeichentexte der GB-Codierung formatiert und ausgedruckt. Nach jedem Satz wurde eine Leerzeile eingefügt, um Sätze beim Lesen durch eine kleine Pause zu trennen. Die Aufnahmen wurden von Wei Tianshi und einer Kollegin in der Volksrepublik China im Juni 1996 durchgeführt. Dazu wurden die präparierten Texte vorgelesen und mit einem Dat-Recorder aufgezeichnet. Die Aufnahme fanden in den 3 Städten Hekou, Wuhan und Peking der Volksrepublik China statt.

Insgesamt wurden 132 Sprecher aufgenommen, wobei jeweils ungefähr eine Hälfte weibliche und die andere männliche Sprecher waren. Alle Sprecher haben zusammen 10214 Sätze mit einer Gesamtlänge von 28,6 Stunden gelesen, was einer durchschnittlichen reinen Lesedauer von ungefähr 13 Minuten pro Sprecher entspricht. Da die Aufnahmen an sehr unterschiedlichen Orten erfolgten, unterscheiden sich auch die Hintergrundgeräusche sehr. Die Bandbreite reicht dabei von völliger ruhiger Umgebung, über Küchengeräuschen, Hintergundgesprächen, bis hin zu Verkehrsgeräuschen.

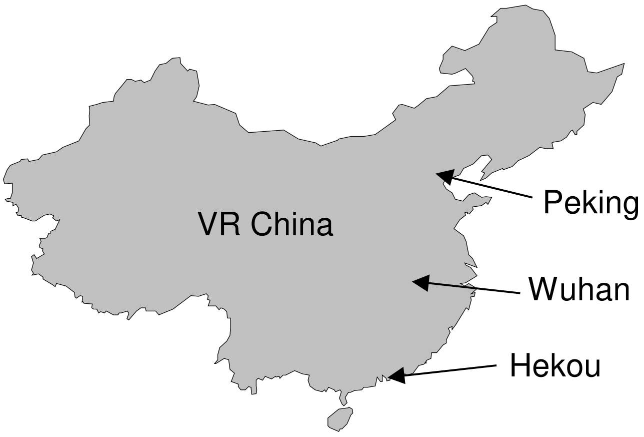


Abbildung 5.1: Aufnahmeorte in der VR China

Durch die verschiedenen Aufnahmeorte wurde auch ein breites Spektrum durch Dialekte eingefärbte Aussprachen überdeckt. Obwohl alle Sprecher angewiesen wurden hochchinesisch zu sprechen, ist eine Beeinflussung ihrer Sprache durch ihren Herkunftsort fast immer gegeben. Im Chinesischen unterscheiden sich die verschiedenen Dialekte auch beim Vorlesen besonders stark, da keine Bindung der Aussprache an die Schrift existiert.

| **Provinz/Stadt** | **Anz** | **Provinz/Stadt** | **Anz** | **Provinz/Stadt** | **Anz** |
|---|---:|---|---:|---|---:|
| Anhui | 3 | Jiangxi | 4 | Sichuan | 1 |
| Chaozhou | 3 | Jiangzhe | 1 | Tianjin | 1 |
| Guizhou | 2 | Kantong | 20 | Wuhan | 6 |
| Hanzhong | 1 | Nanjing | 1 | Yangjiang | 1 |
| Henan | 5 | Beijing | 1 | | |
| Hochchinesisch | 28 | Shanghai | 3 | Sonstige | 22 |
| Hubei | 15 | Shandon | 5 | | |
| Hunan | 8 | Shanxi | 1 | Σ | 132 |

Abbildung 5.2: Zuordnung von Sprechern zu Herkunftsorten

Bei den Aufnahmen wurde Wert darauf gelegt, eine repräsentative Auswahl von Sprechern aus der Bevölkerung zu erhalten. So sollten neben einem ausgewogenen Verhältnis von männlichen und weiblichen Sprechern, alle Altersstufen von 18-60 Jahren gleichmäßig abgedeckt werden. Ein weiterer Punkt auf den geachtet wurde, war, daß die Sprecher aus den unterschiedlichsten Bildungsschichten kamen.

## 5.3 Aufbereitung der Daten

Als nächstes wurden die gesammelten Sprachdaten vom Dat-Recorder auf ein Computersystem digital übertragen, um dort ihre Abtastrate von 48 kHz auf 16 kHz zu verringern. Gleichzeitig wurden beide Stereokanäle zu einem Monokanal zusammengefaßt. Insgesamt verringerte sich dadurch das Datenvolumen um den Faktor 6 auf nunmehr 3,3 GB. In einem nächsten Schritt wurden anhand der Pausen, die jeder Sprecher nach einem Satz zu machen hatte, die Aufnahme in Audiodateien, die genau einen gelesen Satz enthalten, aufgeteilt. Dieser Vorgang mußte manuell überprüft werden, da viele Sprecher die Pause am Satzende oft vergaßen oder auch innerhalb eines Satzes längere Pausen machten. Die so erzeugten Audiodateien wurden für jeden Sprecher in ein eigenes Verzeichnis übertragen, und anhand ihrer Satznummer durchnumeriert.

In einem nächsten arbeitsintensiven Schritt wurden alle Äußerungen von Wei Tianshi und Wang Jing zur Korrektur nochmals mit dem Orginaltext verglichen. Dabei sollten die Orginaltexte durch Korrekturen an die Audiodaten angepaßt werden. Als häufigste Abweichungen traten dabei Wiederholungen, falsche Aussprache, Weglassen von Teilen und vom Leser erzeugte Nebengeräusche wie Husten, starken Atmen, Räuspern und dergleichen auf.

Die in solcher Weise aufbereiteten Texte sollten dann in eine Art Lautschrift umgesetzt werden, um für den Spracherkenner von Nutzen zu sein. Als Umschrift wurde dafür das im Festland China verbreitete Pinyinsystem gewählt. In meiner Studienarbeit „Lautschriftumsetzung und Worttrennung der chinesischen Schriftsprache“ [44] entstand ein solches Umsetzungssystem. Bei der Umsetzung wird anhand statistischer Analysen eine optimale Wortrennung und darauf aufbauend, die beste der oft mehrdeutigen Abbildungen auf die Pinyinumschrift erzeugt.

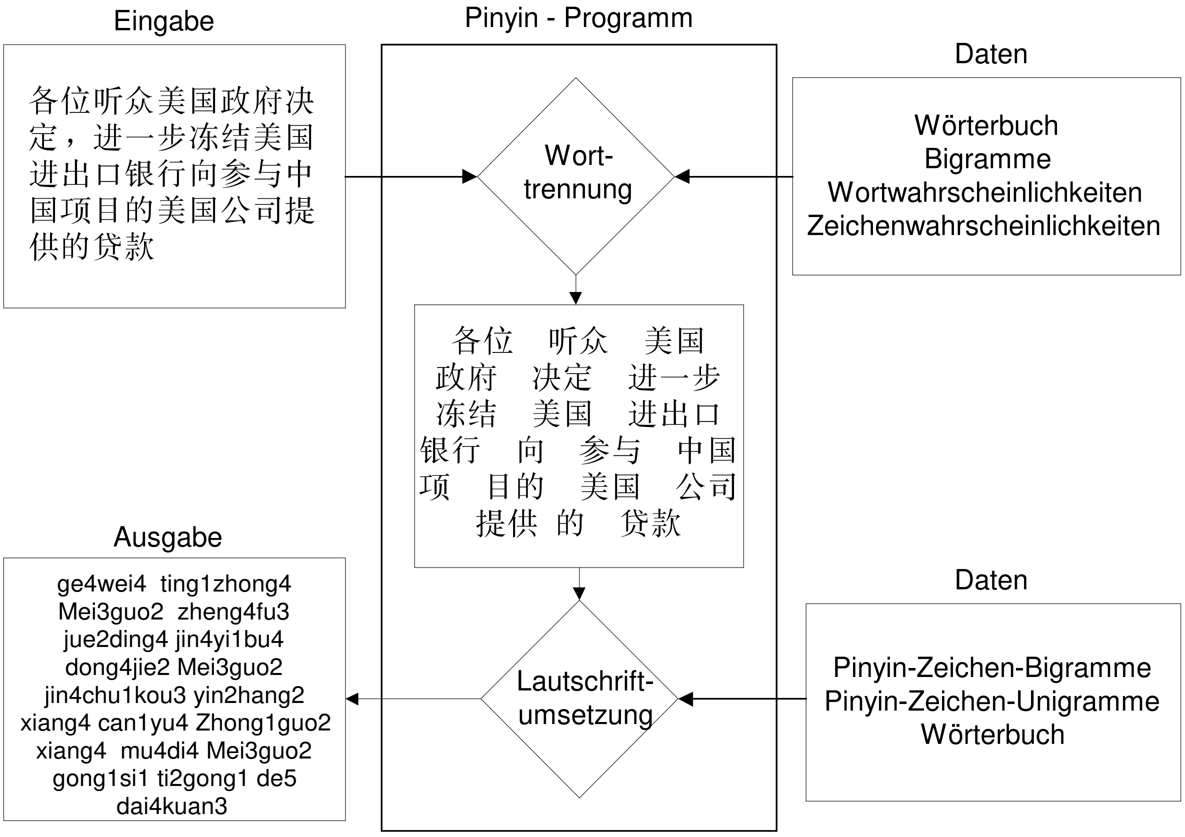
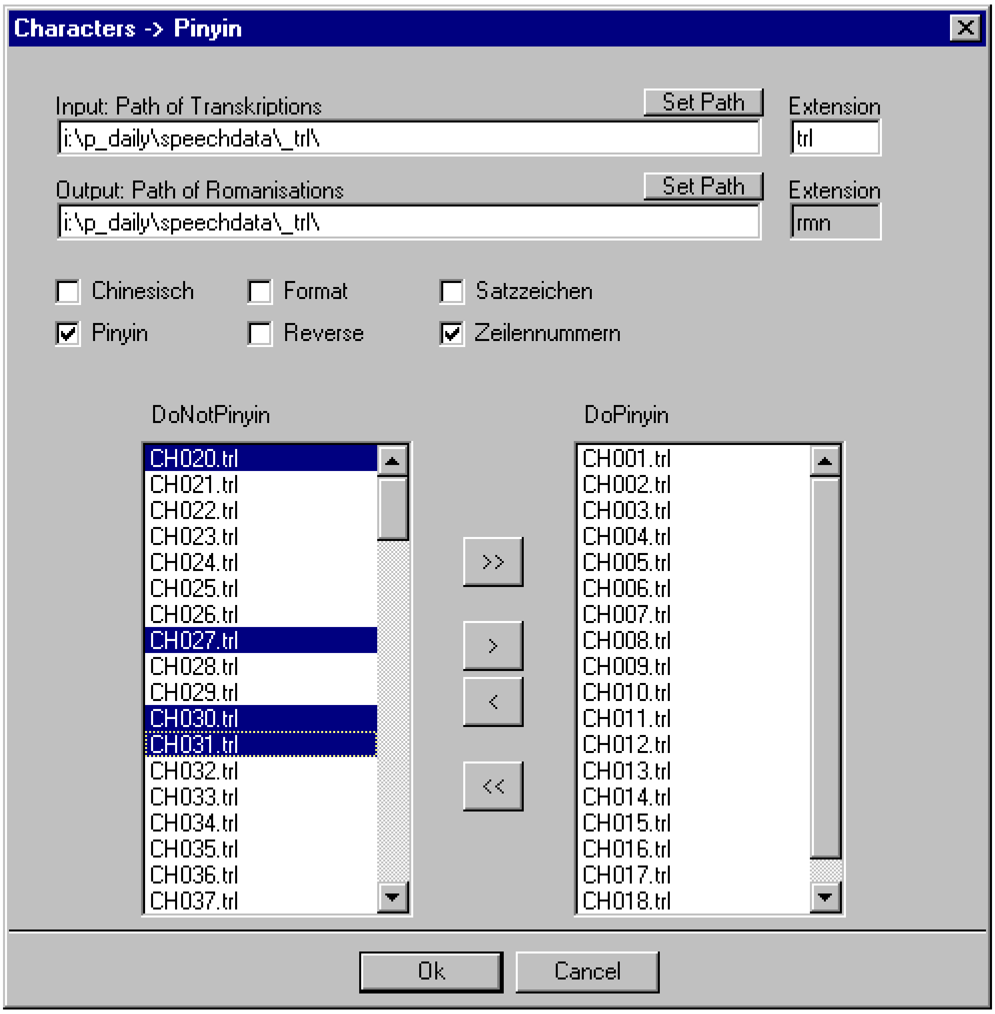

Eingabe
各位听众美国政府决定，进一步冻结美国进出口银行向参与中国项目的美国公司提供的贷款
Pinyin - Programm
Wort-trennung
Daten
Wörterbuch
Bigramme
Wortwahrscheinlichkeiten
Zeichenwahrscheinlichkeiten
各位 听众 美国 政府 决定 进一步 冻结 美国 进出口 银行 向 参与 中国 项 目的 美国 公司 提供 的 贷款
Ausgabe
ge4wei4 ting1zhong4 Mei3guo2 zheng4fu3 jue2ding4 jin4yi1bu4 dong4jie2 Mei3guo2 jin4chu1kou3 yin2hang2 xiang4 can1yu4 Zhong1guo2 xiang4 mu4di4 Mei3guo2 gong1si1 ti2gong1 de5 dai4kuan3
Lautschrift-umsetzung
Daten
Pinyin-Zeichen-Bigramme
Pinyin-Zeichen-Unigramme
Wörterbuch

Abbildung 5.3: Funktionsweise des Pinyinumsetzers

Dieser Pinyinumsetzer wurde im Laufe der Diplomarbeit an vielen Stellen noch optimiert, da seine Qualität einen entscheidenden Einfluß auf das gesamte Erkennungssystem hat. So wurde eine korrekte Umsetzung häufig gebrauchter zusammengesetzter Zahlen implementiert. „５５０“ wird jetzt korrekt in „wu3bai3wu3shi2“ (fünfhundertfünfzig) statt in „wu3wu3ling2“ (fünf fünf null) abgebildet. Genauso wurden Jahreszahlen nun korrekt umgesetzt, die wiederum nicht als Zahl gelesen werde. Z.B. １９９８年 wird jetzt richtig auf „yi1jiu3jiu3ba1nian2“ abgebildet. Ein weiteres Problem stellten die in den Texten häufig verwendeten Prozentzahlen dar, denn im Chinesischen wird das Wort für Prozent vor der Zahl gelesen, obwohl es nach der Zahl geschrieben wird. Z.B. wird aus ８．５％ „bai3fen1zhi1 ba1dian3wu3 (Prozent 8.5)“. Sehr schwierig ist der korrekte Umgang mit Wörtern und Abkürzungen in lateinischer Schrift. Da die meisten Chinesen des Englischen nicht mächtig sind, gibt es sehr viele Varianten, wie solche Wörter wiedergegeben werden. Für die Umsetzung fiel die Entscheidung auf eine Abbildung jedes Buchstabens auf die chinesische Aussprachesilbe, die der englischen Aussprache des Buchstabens am nächsten kam. In einem weiteren Schritt wurde das dem Pinyinprogramm zugrundeliegende Wörterbuch fehlerbereinigt, indem Teile der Aufnahmedaten wiederum mit der Pinyinumsetzung verglichen wurden. Weil die Texte sich mit Innen- und Außenpolitik sowie Wirtschaftsfragen beschäftigen, war eine starke Verwendung von Eigennamen, entweder Politikernamen oder Firmen- und Managernamen festzustellen. Dies machte der Pinyinumsetzung trotz aller Verbesserungen natürlich Probleme, so daß eine Restfehlerrate von ca. 2-3% nicht zu vermeiden war.

Da nach jeder Änderung der chinesischen Texte oder dem Pinyinwörterbuch die Pinyintexte erneut umgesetzt werden mußten, wurde dieser Prozeß vereinfacht, indem das Pinyin-Programm um eine grafische Oberfläche erweitert wurde. Gleichzeitig konnte durch interne Optimierungen die Umsetzungsgeschwindigkeit um den Faktor 10-12 erhöht werden, so daß ein Änderungszyklus nun effizienter durchgeführt werden kann.



Input: Path of Transkriptions: i:\p_daily\speechdata_trl\
Extension: trl

Output: Path of Romanisations: i:\p_daily\speechdata_trl\
Extension: rmn

☐ Chinesisch ☐ Format ☐ Satzzeichen
☑ Pinyin ☐ Reverse ☑ Zeilennummern

| DoNotPinyin | DoPinyin |
|---|---|
| CH020.trl | CH001.trl |
| CH021.trl | CH002.trl |
| CH022.trl | CH003.trl |
| CH023.trl | CH004.trl |
| CH024.trl | CH005.trl |
| CH025.trl | CH006.trl |
| CH026.trl | CH007.trl |
| CH027.trl | CH008.trl |
| CH028.trl | CH009.trl |
| CH029.trl | CH010.trl |
| CH030.trl | CH011.trl |
| CH031.trl | CH012.trl |
| CH032.trl | CH013.trl |
| CH033.trl | CH014.trl |
| CH034.trl | CH015.trl |
| CH035.trl | CH016.trl |
| CH036.trl | CH017.trl |
| CH037.trl | CH018.trl |

Abbildung 5.4: Grafische Oberfläche des Pinyinumsetzers

Beispiel einer Umsetzung eines Satzes:

**Chinesisch**

国民经济保持持续高速发展，原定本世纪末国民生产总值比１９８０年翻两番的目标已提前5年实现，全国大多数人的温饱问题得到解决，社会生产力、综合国力和人民生活上有了一个大台阶。

**Pinyin**

guo2min2jing1ji4 bao3chi2 chi2xu4 gao1su4 fa1zhan3 yuan2ding4 ben3 shi4ji4mo4 guo2min2sheng1chan3zong3zhi2 bi3 yi1jiu3ba1ling2nian2 fan1 liang3 pan1 de5 mu4biao1 yi3 ti2qian2 wu3 nian2 shi2xian4 quan2guo2 da4duo1shu4 ren2 de5 wen1bao3 wen4ti2 de2dao4 jie3jue2 she4hui4 sheng1chan3li4 zong1he2 guo2li4 he2 ren2min2 sheng1huo2shang4 you3 le5 yi1ge4 da4 tai2jie1

## 5.4 Erzeugung der Konfigurationsdateien

Unter den Konfigurationsdateien werden alle zum Aufbau eines Erkenners benötigten Beschreibungen und Daten verstanden (s.a. Kapitel 4). Durch die Erzeugung dieser Daten, die den akustischen, wie auch den sprachlichen Teil des Erkenners definieren, werden nahezu alle

Entwurfsparameter festgelegt. In den nächsten Unterkapiteln werden diese näher, in Bezug zum Aufbau eines chinesischen Spracherkenners, betrachtet.

### 5.4.1 Aufteilung der Daten

Damit Tests aussagekräftig sein können, muß während der gesamten Aufbau- und Trainingsphase des Erkenners darauf geachtet werden, daß keine zum Test verwendeten Daten jemals beim Aufbau oder Training verwendet werden. Sie müssen vollständig aus dem Erzeugungsprozeß ausgeschlossen werden. Eine weitere Unterteilung der ausgeschlossenen Daten ist sinnvoll, so daß Erkenntnisse aus Testläufen dem Erkenner zugute kommen können, ohne dabei eine Evaluation zu beeinflussen. Es entstehen also drei Datenmengen: die Trainingsmenge, die Testmenge und die Evaluierungsmenge. Die Aufteilung der Daten in diese Mengen ist besonders bei wenigen Daten ein problematischer Vorgang, da bei zu kleiner Trainingsmenge der Erkenner nicht optimal trainiert werden kann, oder bei zu kleiner Test- oder Evaluationsmenge die Ergebnisse keine allgemeine Relevanz mehr haben.

Für die Erzeugung chinesischer Spracherkenner wurden die Daten anhand der Sprecher folgendermaßen aufgeteilt.

| **Menge** | **Sprecher** | **Anzahl** |
|---|---|---|
| Trainingsmenge | 1-27, 32-38, 45-79, 90-132 | 112 |
| Testmenge | 28-31, 39-44 | 10 |
| Evaluationsmenge | 80-89 | 10 |
| | | Σ 132 |

### 5.4.2 Definition des Vokabulars

Das zu erkennende Vokabular ist eine für den Spracherkenner wichtige Größe, da es im allgemeinen einen wesentlichen Einfluß auf die Performance eines Erkenners hat. So nimmt nämlich bei einer Vergrößerung auch die Perplexität (ein Maß für den Verzweigungsgrad) zu. Um eine universelle Eingabe durch das Erkennungssystem zu ermöglichen, sollte im Idealfall das Vokabular möglichst groß sein, um eine niedrige „Out of Vocabulary"-Rate (OOV) zu erreichen, was aber andererseits aufgrund höherer Verwechselbarkeit wieder zu einer schlechteren Erkennung führt. Im Falle der chinesischen Sprache ist man in der glücklichen Lage, daß die OOV im Erkennungsprozeß ab einer gewissen Vokabulargröße keine so sehr wichtige Rolle mehr spielt, und zwar deshalb, da einerseits mit ca. 1300 Aussprachesilben der vollständige Sprachumfang abgedeckt ist, und andererseits keine Wortgrenzen in der Ausgabe der chinesischen Zeichen erforderlich sind. D.h. ein nicht bekanntes Wort wird als Aneinanderreihung der Teilworte betrachtet. Die Größe des Vokabulars spielt aber eine wichtige Rolle für die Umsetzungsqualität der Pinyinausgabe des Erkenners in Schriftzeichen. Hier bedeutet ein größeres Vokabular auch eine bessere Umsetzung. Das definierte Vokabular spielt zusätzlich als Lieferant von Kontextinformationen, ähnlich dem Languagemodels, eine wichtige Rolle.

Es erweist sich als erforderlich, daß alle Worte, die in der Trainingsmenge vorhanden sind, ins Trainingsvokabular aufgenommen werden müssen, um Labels (Zuordnung von Audioframes zu Phonemen) schreiben zu können, da bei der Zuordnung von Audiodaten zur Pinyinumschrift Wortgrenzen eine Rolle spielen. Um eine Liste aller Pinyinworte, die in der Trainingsmenge vorhanden sind, zu erhalten, existiert das Perlskript `Z_MakeVovabfromRMN` der Perlskriptesammlung, welches anhand einer Liste aller Trainingssprecher aus den romanisierten Texten eine Vokabularliste als Ausgabe extrahiert. Das Skript `MakeVovabfromRMN` kann per Mausklick in der Perlumgebung ausgewählt werden. Im diesem Skript können, wie auch in allen anderen Perlskripten, die Ein- und Ausgabedateien mittels der Auszeichner „`##@In@`“ bzw. „`##@Out@`“ im Skript direkt angeben werden. Dies bewirkt, daß automatisch die Ein-/Ausgabefenster aktualisiert werden.

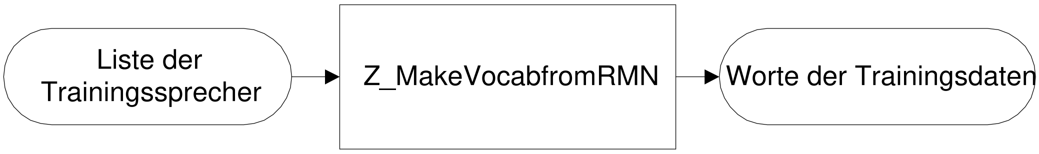


Abbildung 5.5: Erzeugung des Trainingsvokabulars

Für das Testvokabular werden die am häufigsten vorkommenden Wörter aus einem großen Textkorpus extrahiert. Als Textkorpus dient derselbe, der später auch zum Aufbau des Languagemodels benutzt wird. Zuerst wird über die beiden Skripten `LMVocabAnz und` `LMVocabDeckung` eine Statistik über die Abdeckung bei Verwendung von nur mindestens n-mal vorkommenden Worten erstellt.

Beispielausgabe:

```
Mind. 1 Vorkommen
Anzahl Vocab(jetzt): 47207
Deckung %:           100
Anzahl nicht berücksichtigter Wörter:    0
Korpusgröße:         10284392

Mind. 2 Vorkommen
Anzahl Vocab(jetzt): 40415
Deckung %:           99.9339581766234
Anzahl nicht berücksichtigter Wörter:    6792
Korpusgröße:         10284392

Mind. 3 Vorkommen
Anzahl Vocab(jetzt): 36349
Deckung %:           99.8548869004604
Anzahl nicht berücksichtigter Wörter:    10858
Korpusgröße:         10284392

...
```

Über die Angabe einer Mindestanzahl von Vorkommen kann nun die optimale Größe der Ausgabewortliste gesteuert werden.

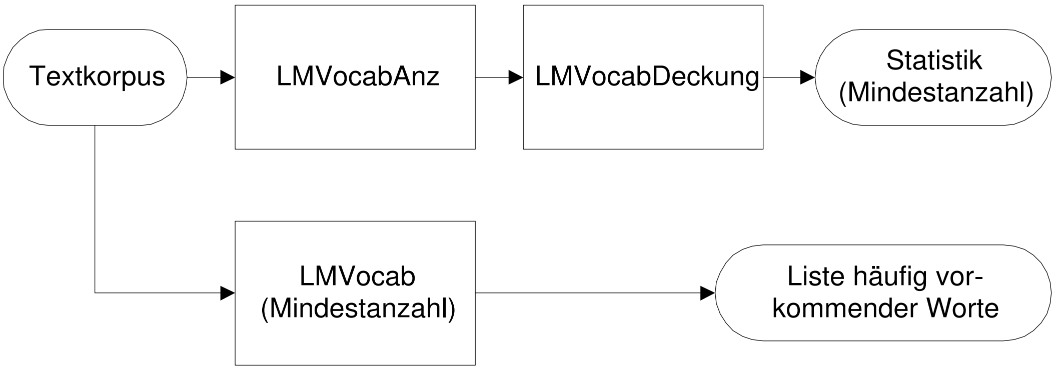

Textkorpus
LMVocabAnz
LMVocabDeckung
Statistik (Mindestanzahl)
LMVocab (Mindestanzahl)
Liste häufig vorkommender Worte

Abbildung 5.6: Erzeugung der Liste häufig vorkommender Worte

Um im Vokabular auch wirklich jede der ca. 1300 Aussprachesilben zu erfassen, so daß jedes unbekannte Wort in Teilworte (Silben) zerlegt werden kann, muß die Menge aller Aussprachesilben erzeugt werden. Hierfür existiert das Perlskript `MakePYListe`, welches anhand einer Liste von Pinyinwörtern als Eingabe alle enthaltenen Aussprachesilben als Ausgabe liefert. Als Pinyinwortliste sollte hier das Gesamtvokabular (Mindestanzahl=1) des Textkorpus verwendet werden.

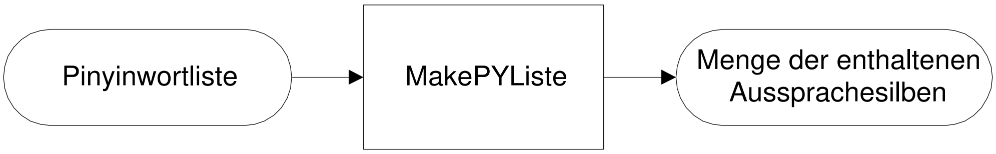

Pinyinwortliste
MakePYListe
Menge der enthaltenen Aussprachesilben

Abbildung 5.7: Erzeugung der Menge aller Aussprachesilben

Die in den obigen Verfahren erstellen Listen müssen nun nur noch zusammengefügt werden, um ein optimales Trainings- bzw. Testvokabular zu erhalten. Das Mischen und Sortieren der Vokabularmengen übernimmt dabei das Perlskript `Z_Join`. Im folgenden wird ein so erzeugtes Trainingsvokabular aus ca. 10000 Worten und ein Testvokabular aus ca. 17000 Worten verwendet.

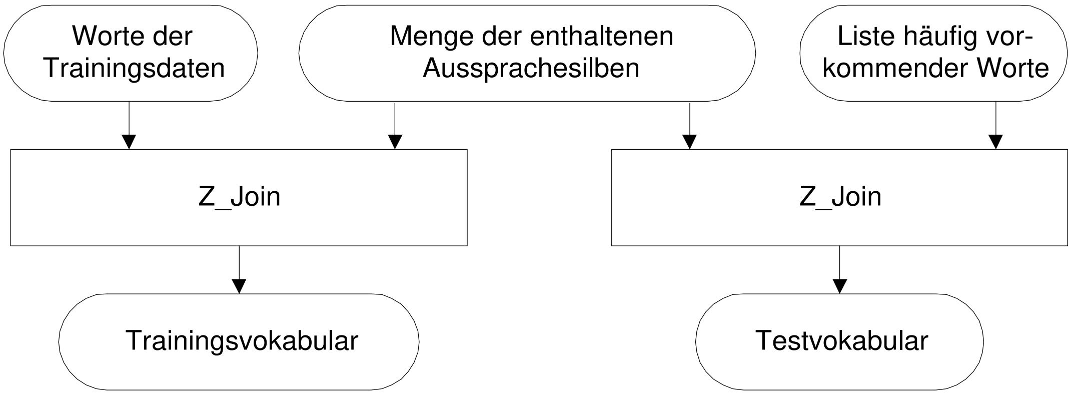

Worte der Trainingsdaten
Menge der enthaltenen Aussprachesilben
Liste häufig vorkommender Worte
Z_Join
Z_Join
Trainingsvokabular
Testvokabular

Abbildung 5.8: Zusammensetzung des Trainings- und Testvokabulars

### 5.4.3 Definition des PhonemSet

Die Pinyinumschrift gibt eine genau Definition der Aussprache chinesischer Zeichen wieder. Allerdings kann die Pinyinumschrift nicht direkt in Phoneme umgesetzt werden, da Laute innerhalb einer Pinyinaussprachesilbe vom direkten Kontext abhängig sind. Um diese Kontextabhängigkeiten aufzulösen wurden spezielle Regeln aufgestellt, die die Pinyinumschrift in eine kontextunabhängige Aussprachedefinition transformieren.

Beispiele solcher Abhängigkeiten:

| | | |
|---|---|---|
| `zi` | `-> z-i` | hinter z wird ein „i“ wie ein sehr kurzes „ö“ gesprochen (-i) |
| `yi` | `-> i` | das „y“ vor Selbstlauten wird nicht gesprochen |
| `wa` | `-> ua` | das „w“ vor einem „a“ wird als „u“ gesprochen |
| `zui` | `-> zuei` | in diesem Kontext wird „ui“ zu „uei“ |
| `dun` | `-> duen` | in diesem Kontext wird „u“ zu „ue“ |
| `qu` | `-> qv` | ein „u“ hinter einem „q“ wird zu „ü“ (als „v“ geschrieben) |
| `juan` | `-> jvan` | in diesem Kontext wird „ua“ zu „üa“ (als „va“) |

Insgesamt lösen 67 solcher Regeln alle internen Abhängigkeiten auf. Um nun aus der Menge der so bereinigten Pinyinaussprachesilben eine Menge von Phonemen zu bilden, müssen diese aufgetrennt werden. Hierfür gibt es natürlich eine große Menge von Möglichkeiten wie das geschehen kann. Ein dazu entwickeltes Perl-Programm kann, über Parameter gesteuert, verschiedene solcher Phonemmengen erzeugen. Dazu trennt es an festlegbaren Stellen die bereinigten Aussprachesilben auf und weist einer definierbaren Teilmenge der Fragmente die Toninformation der Aussprachesilbe zu.

Jede Aussprachesilbe besteht maximal aus einem Anlaut, einem 1.Vokal, einem 2.Vokal, einem 3.Vokal und einem Auslaut. Daraus ergeben sich 4 mögliche Trennstellen, die eine Aussprachesilbe in maximal 5 Fragmente unterteilt. Für jedes Fragment kann entschieden werden, ob ihm die Toninformation der Aussprachesilbe zugewiesen werden soll.

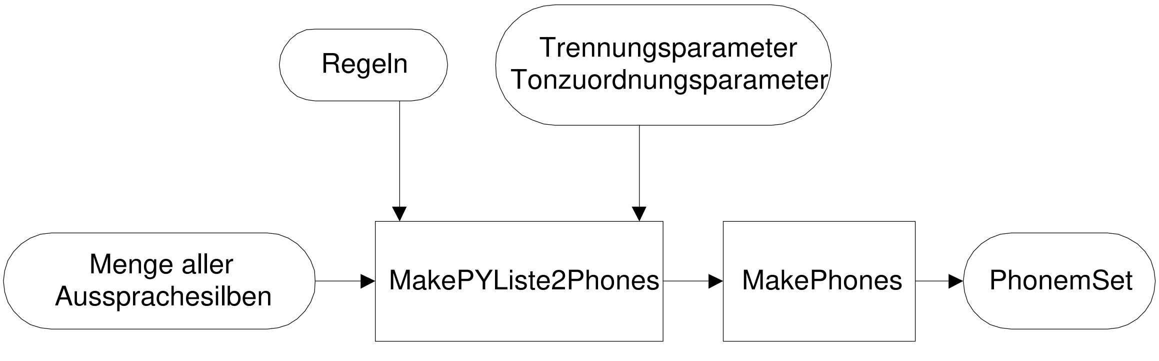


Abbildung 5.9: Erzeugung der Menge aller Aussprachesilben

Nachdem im Skript `MakePYListe2Phones` die Menge der Aussprachesilben anhand der Regeln bereinigt wurde und die Auftrennung der Silben und die Tonzuordnung vorgenommen wurde, werden mittels des Skripts `MakePhones` die einzeln Fragmente (Phoneme) aufgesammelt und die Ausgabe für Janus entsprechend formatiert. Anschließend werden noch die 4

Phoneme „`+QK +hGH @ SIL`“ für die Geräuschklassen 1 und 2, sowie das Padding- und das Silence-Phonem hinzugefügt.

Beispiel für die Erzeugung der Phonemmenge des ersten chinesischen Erkenners:

Trenninformation: 1. und 4. Trennstelle (nach Anlaut und vor Auslaut) -> 3 Fragmente
Tonzuordnung: 2. Fragment

```
sheng3      -> sh|e3|ng
pei4        -> p|ei4
zhuei2      -> zh|uei2
shuang1     -> sh|ua1|ng
iang5       -> ia5|ng
i1          -> i1
...
```

Die Tabelle in Abbildung 5.10 enthält alle 139 so erzeugten Phoneme, sortiert nach den Kategorien Vokale, Diphtonge, Triphtonge und Konsonanten. In der Spalte rechts neben jeder Kategorie befindet sich eine ungefähre Entsprechung im internationalen Phonem Alphabet (IPA 1989) [19].

| **Vokale** | IPA | | **Konsonanten** | IPA |
|---|---|---|---|---|
| -i1 -i2 -i3 -i4 -i5 | I | | b | π |
| a1 a2 a3 a4 a5 | A | | c | τση |
| e1 e2 e3 e4 e5 | Φ | | ch | τη♣ |
| i1 i2 i3 i4 i5 | ι | | d | τ |
| o1 o2 o3 o4 o5 | | | f | φ |
| u1 u2 u3 u4 u5 | υ | | g | κ |
| v1 v2 v3 v4 v5 | Ψ | | h | ξ |
| | | | j | δ♣ |
| **Diphtonge** | | | k | κη |
| ai1 ai2 ai3 ai4 ai5 | Aι | | l | λ |
| ao1 ao2 ao3 ao4 ao5 | AY | | m | μ |
| ei1 ei2 ei3 ei4 ei5 | ει | | n | ν |
| ia1 ia2 ia3 ia4 ia5 | ιA | | ng | N |
| ie1 ie2 ie3 ie4 ie5 | ιE | | p | πη |
| io1 io2 io3 io4 io5 | ιY | | q | τ♣ |
| iu1 iu2 iu3 iu4 iu5 | ιυ | | r | © |
| ou1 ou2 ou3 ou4 ou5 | ο Y | | s | σ |
| ua1 ua2 ua3 ua4 ua5 | υA | | sh | ♣ |
| ue1 ue2 ue3 ue4 ue5 | υE | | t | τη |
| uo1 uo2 uo3 uo4 uo5 | υ | | x | X |
| va1 va2 va3 va4 | ΨA | | z | τσ |
| ve1 ve2 ve3 ve4 | ΨE | | zh | τ♣ |
| | | | | |
| **Triphtonge** | | | | |
| iao1 iao2 iao3 iao4 iao5 | ιAY | | | |
| iou1 iou2 iou3 iou4 | ιYυ | | | |
| uai1 uai2 uai3 uai4 uai5 | υAι | | | |
| uei1 uei2 uei3 uei4 uei5 | υEι | | | |

Abbildung 5.10: Tabelle aller Phoneme mit IPA-Entsprechung

In den folgenden zwei Abbildungen können Entsprechungen der Klassifikationen des IPA-Schemas zugeordnet werden.

| | **Bilabial** | **Labiodental** | **Dental** | **Alveolar** | **Postveolar** | **Retroflex** | **Palatal** | **Velar** | **Uvular** | **Pharyn-geal** | **Glottal** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Plosive** | π β | | τ δ | | | \| | χ<br>\| | κ γ | θ Γ | | ? |
| **Nasal** | μ | M | ν | | | \| | ⎠ | N | ″ | | |
| **Trill** | ⌡ | | ρ | | | | | | { | | |
| **Tab or Flap** | | | P | | | } | | | | | |
| **Fricative** | ÷<br>B | φ ϖ | T<br>Δ | σ<br>ζ | Σ<br>Z | ♣<br>\| | X<br>∅ | ξ<br>⊗ | Ξ<br>® | □<br>א | η \| |
| **Lateral fricative** | | | ℜ Λ | | | | | | | | |
| **Approximant** | | ς | ♦ | | | © | φ | ∑ | | | |
| **Lateral Approximant** | | | λ | | | 〉 | × | ; | | | |
| **Ejective stop** | πэ | | τэ | | | э | χэ | κэ | θэ | | |
| **Implosive** | ≡ | | ⎣ | | | | · | ♥ | } | | |

Abbildung 5.11: IPA-Konsonantenschema
(In der Horizontalen Artikulationsstelle, in der Vertikalen Artikulationsart, links stimmlos/rechts stimmhaft)

| | **Front** | | **Central** | | **Back** |
|---|---|---|---|---|---|
| **Close** | ι ψ | | ⎞ ← | | ∝ υ |
| | | Ι Ψ | | Υ | |
| **Close-mid** | ε Ο | | | | Φ ο |
| | | | ↔ Π | | |
| **Open-mid** | Ε ↵ | | | | ℘ |
| | Θ | | | | |
| **Open** | α ↓ | | | | Α |

Abbildung 5.12: IPA-Vokalschema
(In der Horizontalen steht die Position des Zungenrückens,
in der Vertikalen Mundrundung)

### 5.4.4 Erzeugung des Ausspracheworterbuchs

Die Erstellung des Aussprachewörterbuchs ist in diesem Fall eine einfache Angelegenheit, da beim Erstellen des PhonemSet bereits eine Abbildung aller Aussprachesilben auf die Phoneme erzeugt wurde. So müssen jetzt nur die Worte in Aussprachesilben getrennt werden, die dann in Phoneme umgesetzt werden. In einem weiteren Schritt muß dann die spezielle Formatierung für das Janus-Aussprachewörterbuch vorgenommen werden. Hier spielen die erzeugten Tags, mit denen Phoneme ausgezeichnet werden können, ein wichtige Rolle. In dieser Arbeit wurde als einziger Auszeichner das Wortgrenze-Tag WB (Word boundary) benutzt, das sowohl das Phonem am Wortanfang, als auch das Phonem am Wortende auszeichnet. Mit Tags ausgezeichnete Phoneme werden, in der Phonemliste für jedes Wort, wiederum als Liste aufgefaßt.

Beispiele für korrekt formatierte Einträge im Aussprachewörterbuch:

```
{SIL}                          {{SIL WB}}
a1er3ba1ni2ya4                 {{a1 WB} er3 ba1 ni2 {ia4 WB}}
a1gen1ting2                    {{a1 WB} gen1 {ting2 WB}}
di4                            {{di4 WB}}
di4er4ci4shi4jie4da4zhan4      {{di4 WB} er4 c-i4 sh-i4 jie4 da4 {zhan4 WB}}
huang2huang2bu4ke3zhong1ri4    {{huang2 WB} huang2 bu4 ke3 zhong1 {r-i4 WB}}
...
```

Das Perlskript MakeDict kann anhand des Vokabulars und der Abbildung der bereinigten Aussprachesilben auf die Phoneme das korrekt formatierte Aussprachewörterbuch erzeugen.

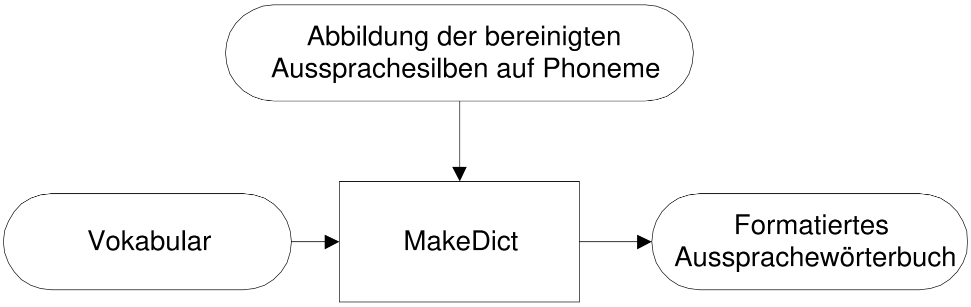


Abbildung der bereinigten Aussprachesilben auf Phoneme
Vokabular
MakeDict
Formatiertes Aussprachewörterbuch

Abbildung 5.13: Erzeugung des Aussprachewörterbuchs

Für den Aufbau des chinesischen Spracherkenners wurden keine Aussprachevarianten ins Wörterbuch aufgenommen, da solche, wenn man Dialekte und englische Fremdwörter außer acht läßt, praktisch sehr selten vorkommen.

### 5.4.5 Konstruktion des Languagemodels

Um die Suche in der Erkennungsphase wirkungsvoll einschränken zu können, muß der akustische Teil des Spracherkenners um den sprachlichen Teil, das Language Model, ergänzt werden. Dies geschieht dadurch, daß zu dem akustischen Score AM ein sprachlicher Score hinzugefügt wird. (lz gewichtet das Languagemodel, während lp zu kurze Worte bestraft)

$$\text{Wordscore} = \text{AM(word)} + \text{lz} * \log_{10}(\ P(\text{word}|\text{history})\ ) + \text{lp}$$

Der zur Erstellung des chinesischen Language Models notwendige Textkorpus wurde aus einer HTML-Textsammlung der chinesischen Tageszeitung „People's Daily“, auf deren Textgrundlage auch die Sprachaufnahmen erfolgten, und der Zeitung „XinHua (Neues China)“ extrahiert. Zur Wandlung der HTML-Dateien in reine Texte wurde das Perlskript `Y_FormatText` erstellt. Dieses beseitigt alle HTML-Steueranweisungen und Strukturinformationen wie Überschriften, Datum, Quellenangaben. Innerhalb des Skriptes mußten verschiedene HTML-Formate berücksichtigt werden, da sich diese teilweise leicht änderten. Die chinesischen Texte wurden dann durch den Pinyinkonverter in die Pinyinumschrift konvertiert.

Aus der Zeitung „People's Daily“ wurden die Jahrgänge 1991-96 und aus der Zeitung „XinHua“ die Jahrgänge 1994-96 verwendet. Dabei wurden die zur Sprachaufnahme verwendeten Texte der Test- und Evaluationsmenge ausgeschlossen.

| | **Zeichen** | **Pinyin** | **Anzahl Worte** |
|---|---:|---:|---:|
| **Peoples‘ Daily** | 295,9 MB | 527,4 MB | 66.069.605 |
| **Xinhua** | 63,2 MB | 98,6 MB | 16.515.689 |
| Σ | 359,1 MB | 625,9 MB | 82.585.294 |

Die mittlere Wortlänge einschließlich der Toninformation beträgt somit ca. 6,6 Zeichen
(625,9 Mio / 82,5 Mio = 7,6 Zeichen inkl. Leerzeichen).

Zum Erzeugen des Language Models wurde das Tool `ngrammodel` von Klaus Ries [46] verwendet. Dieses Tool erzeugt aus einem Textkorpus ein statistisches Language Model, indem es die Wahrscheinlichkeit des Auftretens von Wortfolgen bestimmt. Neben den Wahrscheinlichkeiten von Unigrammen, Bigrammen und Trigrammen wird auch der Backoffaktor für den Rückfall von Trigrammen auf Bigramme und von Bigrammen auf Unigramme berechnet. Ein Rückfall auf ein n-Gramm mit kleinerem Kontext ist immer dann nötig, wenn über einen Kontext keine sicheren Informationen vorliegen. Die Backoffaktoren sind Korrekturwerte, um die einzelnen Wahrscheinlichkeiten miteinander verrechnen zu können. Sie werden nach einem Verfahren von Kneser/Ney bestimmt.
Das Tool `ngrammodel` verwendete den 82,5 Mio. Worte umfassenden Pinyin-Textkorpus zur Erzeugung des Languagemodels. Dabei wurden nur n-Gramme, die mindestens drei mal vorkamen, aufgenommen (Cutoff = 3) .

| | Anzahl n-Gramme | Überdeckung |
|---|---:|---:|
| **Trigramme** | 4086172 | 46.71% |
| **Bigramme** | 1815298 | 39.53% |
| **Unigramme** | 45082 | 13.76% |

Die Perplexität des Languagemodels beträgt 207,5.

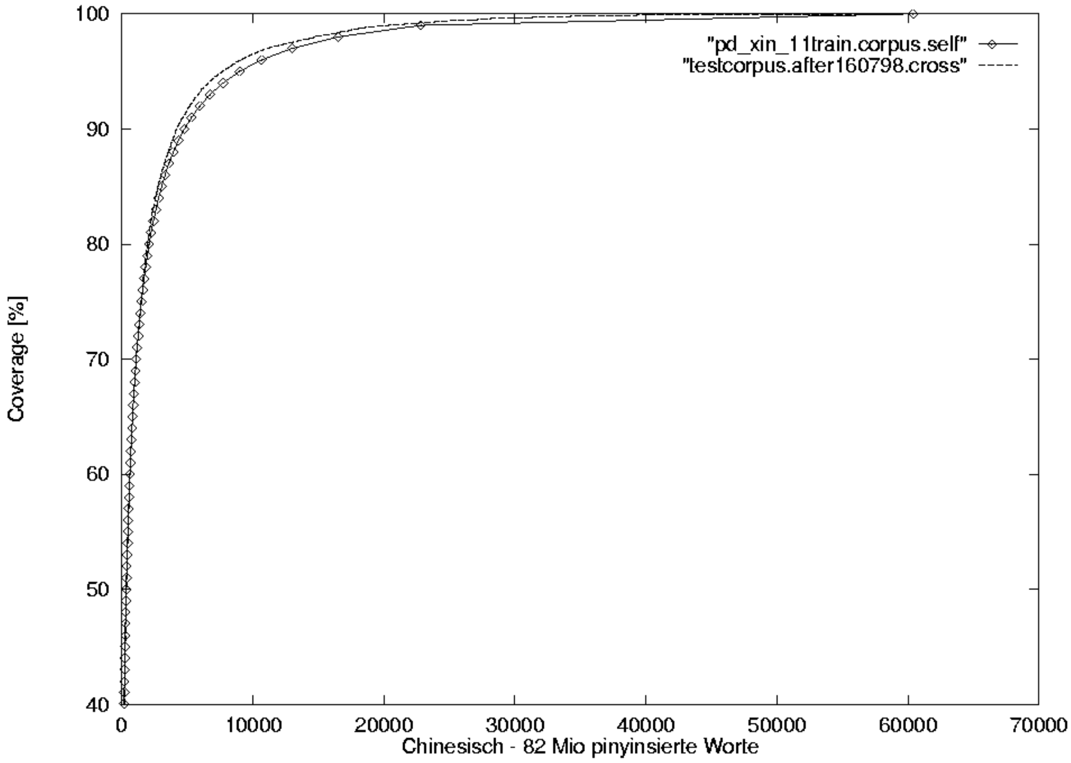


Abbildung 5.14: Überdeckung des Korpus durch die n-häufigsten Worte

#### 5.4.5.1 Konstruktion von Language Models unter Windows

Da das Tool `ngrammodel` nur auf Unix-Workstations lauffähig ist, mußte, um eine vollständige Entwicklungsumgebung für Spracherkennungssysteme unter Windows zu erreichen, eine Möglichkeit gefunden werden, auch unter dem Windows-Betriebsystem Languagemodels erzeugen zu können. Diesmal wurde nicht der Weg der Portierung gewählt, sondern eine stark eingeschränkte Version des Tools neu entwickelt. Dies sollte zumindest ermöglichen, angemessene Sprachmodelle während der Konstruktionsphase eines Spracherkenners zur Verfügung zu stellen.

Bei dem Tool `MakeLM` handelt es sich um ein Perlskript, welches ein NIST-konformes Language Model erzeugen kann. Das Tool ist auf Trigramme beschränkt und unterstützt keine Wortklassenbildung. Die Backoffaktoren werden anhand einer etwas vereinfachten Variante berechnet. Das Tool ist nicht auf Geschwindigkeit optimiert, so daß auf Standard-PCs ein Textkorpus von 20-30 Mio. Worte nicht überschritten werden sollte, außer man hat genügend

Zeit. Die Hauptspeichergröße hat einen direkt Einfluß auf die Ausführungsgeschwindigkeit, eine Mindestanforderung existiert aber nicht. Der Festplattenplatz für temporäre Dateien sollte großzügig bemessen sein (ca. 3-fache Korpusgröße).

Dem Tool `MakeLM` kann, alternativ zu Cutoff-Faktoren, ein Vokabular angegeben werden, so daß nur n-Gramme, die aus Wörtern dieses Vokabulars bestehen, ins Languagemodel aufgenommen werden. Dies hat den Vorteil, wenn der Textkorpus nicht so gut zum Task paßt, daß nur die relevanten n-Gramme aufgenommen werden. Über das Skript `LMVocabDeckung` kann die Abdeckung des Textkorpus vom Vokabular bestimmt werden, wenn nur die Wörter verwendet werden, die mindestens n mal vorkommen.

Das `MakeLM-Tool` scannt den Textkorpus, je nach Größe des vorhandenen Hauptspeichers, jeweils für einen bestimmten Teil des Vokabulars. Alle n-Gramme, deren letztes Wort in diesem Vokabularteil vorhanden ist, werden gezählt, und für jeden Teil des Vokabulars auf Platte geschrieben. Um eine geringe Anzahl von Durchläufen durch den Textkorpus zu erhalten, wird der Hauptspeicher optimal genutzt, indem Vokabular und n-Gramme indiziert werden. Im nächsten Schritt werden die Backoffaktoren und logarithmierten Wahrscheinlichkeiten berechnet.

$$P(w_n \mid w_{n-1}..w_1) = \begin{cases} q(w_n \mid w_{n-1}..w_1) & ,\mathit{falls}\ \mathit{Anz}(w_n w_{n-1}..w_1) > 0 \\ \mathit{Backoff}(w_{n-1}..w_1) \cdot P(w_n \mid w_{n-1}..w_2) & ,\mathit{falls}\ \mathit{Anz}(w_n w_{n-1}..w_1) = 0 \end{cases}$$

Die Wahrscheinlichkeiten ergeben sich dabei nach obiger rekursiver Formel. Mit q:

$$q(w_n \mid w_{n-1}..w_1) = \frac{f\left(\mathit{Anz}(w_n w_{n-1}..w_1)\right)}{\mathit{Anz}(w_{n-1}..w_1)}$$

Die Funktion $q$ ist die mittels $f$ verringerte Auftrittsanzahl von $w_n w_{n-1}..w_1$. Die so eingesparte Wahrscheinlichkeitsmasse wird mittels der Backoffaktoren auf die ungesehenen n-Gramme verteilt. Die Qualität von Backoff-Verfahren wird durch die Bestimmung dieser Funktion $f$ maßgeblich beeinflußt. Dabei ist ein Kompromiß zwischen Qualität und Berechnungsaufwand zu finden. Bei dem hier implementierten Verfahren wurde die umzuverteilende Wahrscheinlichkeit anhand der Anzahl nur einmal vorkommender n-Gramme oder n-Grammen, deren Worte nicht im Vokabular vorkommen, abgeschätzt. Der Backoffaktor stellt sicher, daß die Wahrscheinlichkeitssumme weiterhin 1 ergibt und kann bei Kenntnis der Wahrscheinlichkeiten der (n-1)-Gramme einfach berechnet werden.

In einem weiteren Durchlauf wird aus den so erhaltenen statistischen Daten ein NIST-konformes Languagemodel erstellt. Das NIST-Format enthält einen Header, in dem die jeweilige Anzahl der n-Gramme angegeben ist. Dann folgt ein Block für Unigramme, Bigramme und Trigramme, in denen pro Zeile ein n-Gramm mit logarithmierter Wahrscheinlichkeit und Backoffaktor aufgeführt ist.

Auszug aus NIST-konformen Language Model:

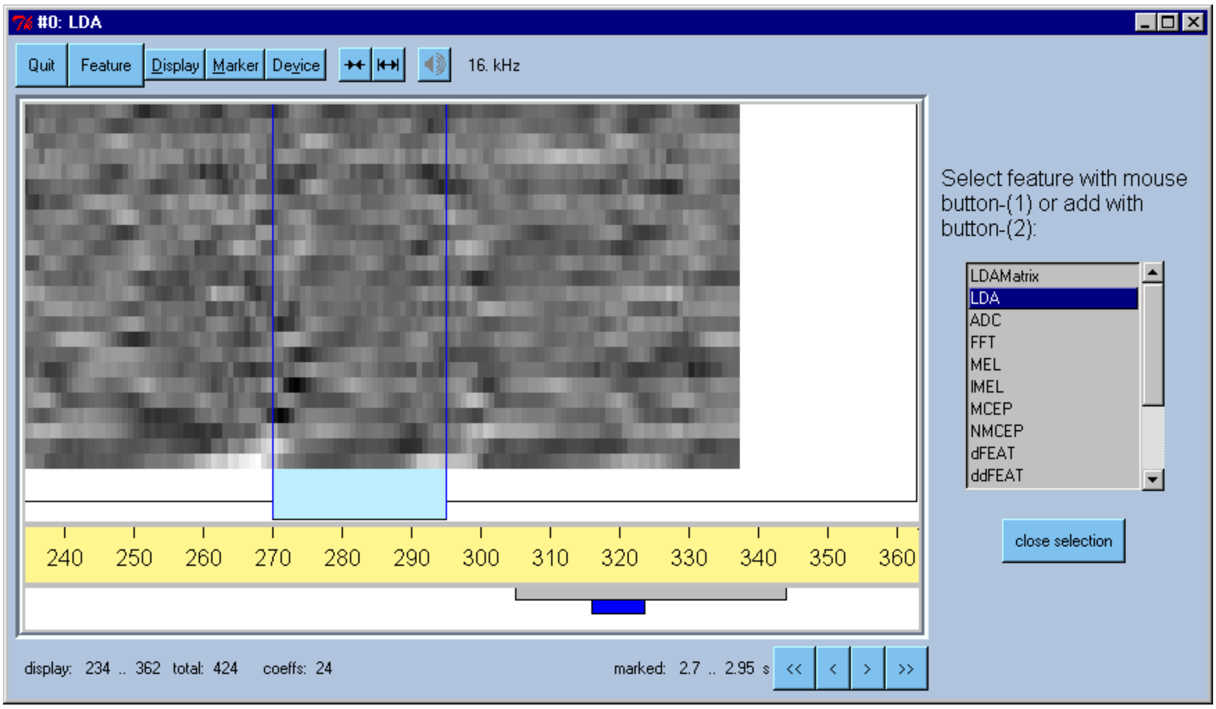
```
\data\
ngram 1=47210
ngram 2=1983515
ngram 3=5363681

\1-grams:
-3.72074991839063 <s> 0
-4.8877245806824 a1 -0.314818280149614
-5.0064143680137 a1bu4zha1bi3 -0.823908740944319
-4.98228068829698 a1er3ba1ni2ya4 -1.07188200730613
-1.28630673884327 an4ji4hua2 zu3zhi1 -0.477121254719663

\2-grams:
-1.83686980358 zuo4yong4 fa1zhan3zhong1guo2jia1 -0.923908740944319

\3-grams:
-0.933672084903 zuo4chu1 gong4xian4 chan3sheng1
\end\
```

### 5.4.6 Vorverarbeitung und HMM

Um Festplattenplatz einzusparen, können die Audiodaten auf Festplatte, auf Kosten geringer Geschwindigkeitsverluste, komprimiert werden. Die Kompression ist auch bei Sicherung oder Verteilung der Audiodaten per Datenträger sinnvoll. Dazu eignen sich besonders speziell für Audiodaten optimierte Kompressionsverfahren. SHORTEN von Tony Robinson [47] erreicht eine hohe, verlustfreie Kompressionsrate auf Audiodaten durch ein LPC-Verfahren mit anschließender Huffmanncodierung. Janus ist in der Lage mit solchen Daten umzugehen, indem in der Feature-Description-Datei das „-bm“-Flag gesetzt wird. Das Perlskript `Z_Shorten` kann unter Windows die Konvertierung der Audiodaten aller Sprecher durchführen. Gegenüber Winzip, daß nur eine Komprimierung von ca. 15 % erreicht, liegt die Komprimierungsrate von SHORTEN bei ca. 27 %.

Die Vorverarbeitung der Audioinformationen umfaßt die Schritte vom PCM-Signal bis zu den Merkmalsvektoren. Alle Vorverarbeitungsschritte sind Methoden des FeatureSet-Objekts. Mittels „readADC“ wird das 16 Bit, 16 kHz abgetastete PCM-Mono-Signal eingelesen und dann durch die Methode „spectrum“ in den Frequenzbereich überführt. Dann werden 30 melscale-Filterbankkoeffizienten gebildet. Diese werden dann logarithmiert und in 13 normierte Cepstren transformiert. Zusätzlich zu den normierten Cepstren gehen deren Differenzen zu den Nachbarframes, sowie die Differenzen der Differenzen in den Merkmalsvektor ein. Außerdem werden noch das Zerocrossing, die Leistung, die Leistungsdifferenzen, sowie die Differenzen der Leistungsdifferenzen zum Merkmal hinzugefügt. Es ergeben sich somit 43 (3*13+1+1+1+1) Komponenten, die einen 43 dimensionalen Merkmalsraum aufspannen. Die lineare Diskriminanzanalyse bildet anhand der Trainingsdaten daraus 24 dimensionale Merkmalsvektoren, die die wichtigsten Informationen zur Unterscheidung der Phonemzustände enthalten.

Diese Vorverarbeitung wurde gewählt, um mit anderen Spracherkennern aus dem “GlobalPhone”-Projekt konform zu sein. Andere Vorverarbeitungsverfahren, die die Tonalität der chinesischen Sprache evtl. besser erfassen, sind sicher eine weitere Untersuchung wert.


#0: LDA
Quit
Feature
Display
Marker
Device
16. kHz
240
250
260
270
280
290
300
310
320
330
340
350
360
LDAMatrix
LDA
ADC
FFT
MEL
lMEL
MCEP
NMCEP
dFEAT
ddFEAT
close selection
<<
<
>
>>

Select feature with mouse button-(1) or add with button-(2):

display: 234 .. 362 total: 424 coeffs: 24 marked: 2.7 .. 2.95 s

Abbildung 5.15: Feature-Betrachter aus der Janus Toolssammlung

Als HMM-Topologie wurde eine Vorwärts-Topologie vom Grad 1 gewählt. Das heißt es gibt nur Selbstübergänge und Übergänge in den direkten Nachfolger. Für jedes Phonem wurden 3 Zustände im HMM definiert. Dies ist ein Kompromiß zwischen einer großen Anzahl von Zuständen, die den Tonverlauf im Chinesischen gut berücksichtigt und einer nicht zu großen Anzahl, um alle Zustände rechtzeitig durchlaufen zu können, da Chinesisch teilweise sehr schnell gesprochen wird. Das Silence-Phonem wurde nur durch einen Zustand modelliert.
Die Ausgabewahrscheinlichkeiten der HMM-Zustände wurden durch eine Mischung von 16 Gaußverteilungen modelliert, deren Kovarianzmatrizen nur auf der Diagonalen besetzt sind. In der folgenden Abbildung werden exemplarisch zwei solcher Mischungen von Gaußverteilungen herausgegriffen.

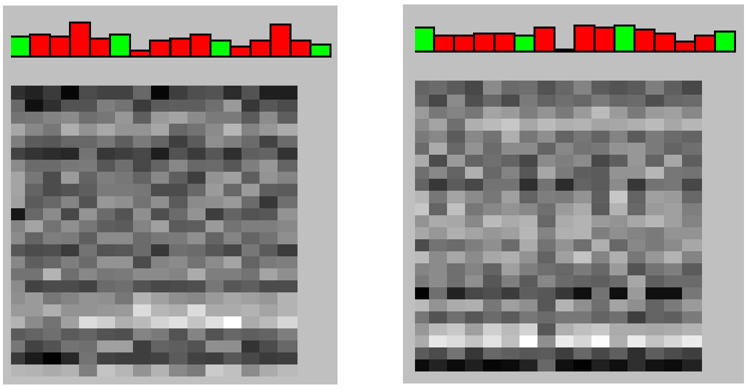


Abbildung 5.16: Codebuch/Distribution-Betrachter aus der Janus Toolssammlung
(oben die 16 Verteilungsgewichte, unten die 16 Mittelwerte der Dimension 24)
(links Vokal ao1-m, rechts Konsonant b-m)

### 5.4.7 Initialisierung des chinesischen Erkenners

Wie im 4. Kapitel erläutert, wird auch der chinesische Erkenner anhand eines schon bestehenden Erkenners initialisiert. Zur Anwendung kommt der multilinguale Erkenner, der für die Sprachen Deutsch, Englisch, Japanisch und Spanisch trainiert wurde. Dazu wird eine Mapping-Tabelle aufgestellt, die jedem Phonem des chinesischen Erkenners ein Phonem des multilingualen Erkenners zuordnet. Für die tonale Zuordnung stehen keine Vorlagen zur Verfügung, so daß alle tonalen Varianten eines Phonems dieselbe Initialisierung erhalten.
Diese Zuordnung muß von Hand nach größtmöglicher Ähnlichkeit erfolgen. Die Qualität ist in zweierlei Hinsicht wichtig, zum einen ist sie ausschlaggebend dafür, wie schnell der Erkenner ein lokales Optimum der Performance erreicht, und zum anderen ist der Initialisierungszustand entscheidend dafür, welches lokale Optimum erreicht wird, also wie gut der Erkenner am Ende wird.
Zur Optimierung und Überprüfung dieser Zuordnung wurden zwei spezielle Tcl-Skripts entwickelt. `Score1.tcl` berechnet den Score einer Äußerung, so daß verschiedene CodebookSets und DistribSets miteinander verglichen werden können. Das Skript `Score2.tcl` berechnet für jeden Frame die n wahrscheinlichsten Phoneme und gibt sie nach Wahrscheinlichkeit sortiert aus. Auf diese Art und Weise können auch Probleme der Akustik aufgespürt oder einzelne Phoneme gezielt verändert werden.

## 5.5 Training des kontextunabhängigen Erkenners

Nachdem alle Konfigurationsdateien erzeugt wurden, kann mit dem eigentlichen Training begonnen werden, bei dem die Trainingsphasen „Labels schreiben“, „LDA berechnen“, „Samples extrahieren“, „Kmeans berechnen“, „Training“ mehrfach iterativ durchlaufen werden. Bei jeder Iteration geht man davon aus, daß nun der Erkenner bessere Labels schreibt und durch die besseren Labels der neue Erkenner noch besser wird. Allerdings beansprucht eine solche Iteration auf einem einzigen Standard-PC mehrere Tage Rechenzeit. Somit ist es sinnvoll mehrere Prozessoren oder Rechner gleichzeitig die Berechnungen ausführen zu lassen. Janus wurde dazu auf Tcl-Basis um ein Parallelkonstrukt erweitert, das es erlaubt, Berechnungen auf mehreren Prozessoren oder Rechnern parallel auszuführen. Dazu werden jeweils die Berechnungen, die genau einen Sprecher (Phonem, Turn) betreffen, von dem nächsten freien Prozeß ausgeführt.

Das gesamte Training des chinesischen Spracherkenners wurde auf zwei PCs mit jeweils einem Prozessor mit 200 bzw. 266 MHz und jeweils 128 MB Hauptspeicher parallel ausgeführt. Eine Iteration mit anschließendem Performancetest dauerte ca. 2-3 Tage reine Rechenzeit.

### 5.5.1 Labels schreiben

In der ersten Trainingsphase werden für alle Äußerungen Labels geschrieben. Labels sind Zuordnungen der Frames zu Sub-Phonemen. Diese Zuordnungen sind wichtig, damit der Spracherkenner während des Trainings weiß, was für ein Phonem gerade gesprochen wird. Diese Zuordnung kann natürlich auch von Hand erfolgen, indem man für jedes Phonem den Start- und Endframe festlegt. Die manuelle Festlegung kann aber wegen des enormen Aufwandes nur für sehr wenige Äußerungen durchgeführt werden. Deshalb benutzt man einen vorhandenen, wenn auch nicht unbedingt optimalen Erkenner der diese Zuordnung immerhin ansatzweise vornehmen kann. In jeder Iteration sollte dann diese Zuordnung besser werden. Da der Erkenner weiß, was gesprochen wurde, muß er eigentlich keine wirkliche Erkennung durchführen, sondern nur festlegen, zu welchem Zeitpunkt welches Phonem gesprochen wurde. Dazu eignet sich der Viterbi-Algorithmus, der eine optimale Zuordnung findet.

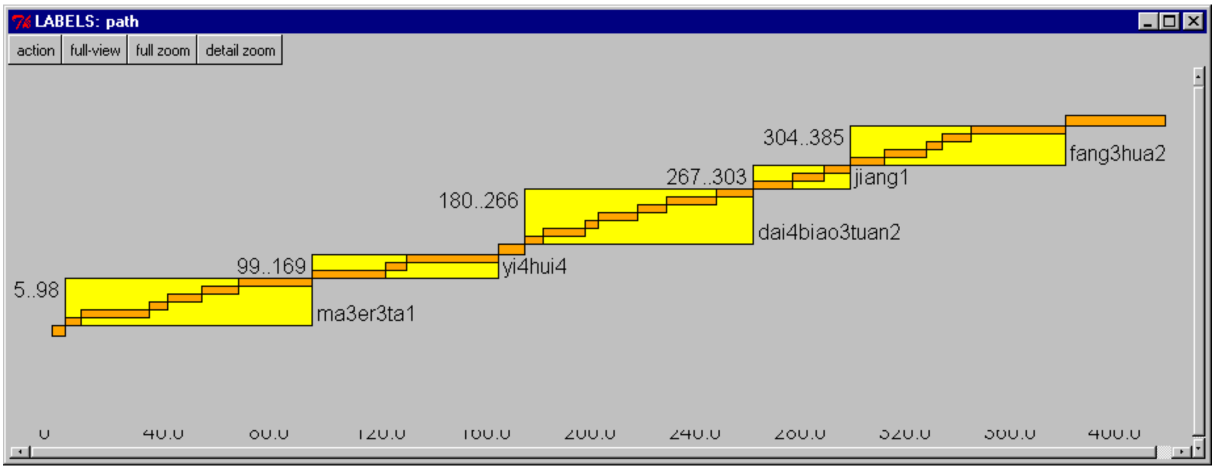


Abbildung 5.17: Zuordnungspfad-Analyse aus der Janus Toolssammlung

Diese Zuordnungen werden in Labeldateien abgelegt, so daß in den weiteren Trainingsphasen die Viterbipfade nicht erneut berechnet werden müssen. Die Analyse von solchen Zuordnungspfaden kann wichtige Informationen über Probleme des Erkenners liefern und sollte daher immer überprüft werden. Dazu existiert aus der Janus Toolssammlung ein Tool zur Zuordnungspfad-Analyse, mit dem bis auf HMM-Zustandsebene herunter die Zuordnung verfolgt werden kann.

### 5.5.2 LDA berechnen

Das Ziel der linearen Diskriminanzanalyse ist, wie in Kapitel 2 beschrieben, die Reduzierung der Dimensionalität des Merkmalsraumes bei Beibehaltung aller wichtigen Informationen und eine bessere Trennbarkeit unterschiedlicher Klassen durch Verringerung der klasseninternen Varianzen. Man erhält eine Matrix, mit der, im Falle des chinesischen Spracherkenners, der 43 dimensionale Merkmalsvektor multipliziert wird, und so in einen 24 dimensionalen Merkmalsvektor übergeht. Die LDA-Matrix ist vom akustischen Modell abhängig und sollte bei dessen Änderung auch neu berechnet werden.

### 5.5.3 Samples extrahieren

Anhand der Labels und der neuen LDA kann jetzt für jeden Frame aller Trainingsdaten jeder Merkmalsvektor genau einem Subphonem zugeordnet werden. Die Anzahl der so aufgesammelten Beispielvektoren wird auf eine Maximalanzahl beschränkt und für jedes Subphonem separat in eine Datei geschrieben.

### 5.5.4 Kmeans berechnen

Aus den gesammelten Beispielvektoren werden mittels des Kmeans-Verfahrens für jedes Subphonem die initialen 16 Gaußverteilungen berechnet. Im ersten Schritt wird dazu ein Clustern in 16 Klassen vorgenommen, indem iterativ jeweils jeder Beispielvektor dem nächsten Mittelpunktsvektor zugeordnet wird und dann 16 neue Mittelpunktsvektoren berechnet werden, solange bis keine Änderungen mehr auftreten. In der Realisierung dieses Verfahrens wird allerdings jeder Beispielvektor zu jeder Klasse anteilsmäßig, aber abhängig zur Entfernung zu dessen Mittelpunktsvektor zugeordnet, um die Klassenverteilung gleichmäßiger durchzuführen. (Sonst bilden evtl. einzelne, abgelegene Vektoren zuviele eigene Klassen). Desweiteren wird die Entfernungsfunktion mit einer über die Zeit zunehmenden „Temperatur“ beaufschlagt, um später nur noch nahegelegene Vektoren zu berücksichtigen (dieses Verfahren wird auch als „neural gas“-Verfahren bezeichnet).
Im nächsten Schritt werden aus dieser Clusterung für jede Klasse der Mittelpunktsvektor und die Kovarianzmatrix bestimmt, aus denen dann das Codebook gebildet wird.

### 5.5.5 Trainingslauf

Beim Training werden durch eine Art Forward-Backward-Algorithmus oder Viterbi-Algorithmus die Streamparameter (Codebooks/Distributions) und die Zustandsübergangswahrscheinlichkeiten optimiert (s. Kapitel 2). In der Praxis ist es aber zu aufwendig die Zustandsübergangswahrscheinlichkeiten mitzubetrachten. Experimente haben ergeben, daß diese Vereinfachung nahezu keine Verschlechterung mit sich bringt. Der Trainingspass wird im allgemeinen mehrmals direkt hintereinander ausgeführt, wobei in jeder Iteration die alten Streamparameter durch die neuen ausgetauscht werden.

### 5.5.6 Test

Nach jedem Training, bevor mit dem neuen Erkenner wieder Labels geschrieben werden, kann und sollte die Performance des Erkenners überprüft werden.
Dazu können für die Suche die Werte folgender Parameter festgelegt werden:

```
Beam: maximaler Abstand des Scores zum besten Suchpfad
TopN: maximale Verzweigungsgrad des Suchpfads nach einem Wortende
lmWeight: Gewichtung des Language Models
lmPenalty: Wortübergangsstrafe
lmWeightList: Liste Gewichtungen beim Rescoring
lmPenaltyList: Liste Wortübergangsstrafen beim Rescoring
```

Die Suche verläuft in 3 Durchgängen. Zuerst wird eine Viterbisuche durchgeführt, die einen Suchbaum zurückliefert (Tree pass). Dieser Suchbaum liefert eine erste Hypothese der Äußerung. Um diese Hypothese in einem zweiten Pass (flat-forward) zu verfeinern, werden in einem Rückwärtsdurchlaufen des Suchbaums (Backtrace) eine Liste der Worte gebildet, die mit hoher Wahrscheinlichkeit in der Äußerung vorkamen, so daß der bei erneuter Viterbisuche aufgebaute Baum nun viel flacher ausfällt, da viele potentielle Irrwege vermieden werden. Im dritten Pass (lattice rescoring) wird aus dem Suchbaum eine Lattice (Trellis) gebildet, in dem entsprechende Knoten im Suchbaum zusammengefaßt werden. Auf dieser Latice können durch Rescoring mit jeweils den Gewichten aus der `lmWeightList` und den Wortübergangsstrafen aus der `lmPenaltyList` eine große Menge von Hypothesen für jede Äußerung gebildet werden.

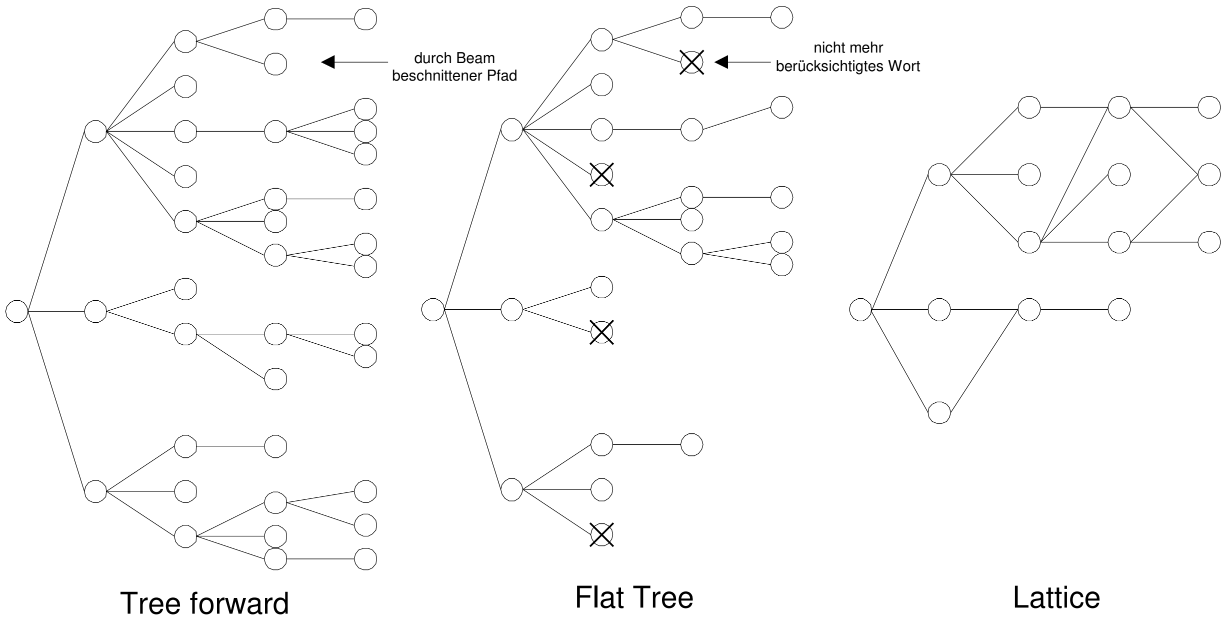

durch Beam beschnittener Pfad
nicht mehr berücksichtigtes Wort
Tree forward
Flat Tree
Lattice

Abbildung 5.18: Veranschaulichung der verschiedenen Suchpässe

#### 5.5.6.1 Optimierung der lm-Parameter

Die große Menge der durch Rescoring gebildeten Hypothesen werden dazu verwendet, um die optimalen Werte für die Parameter `lmWeight` und `lmPenalty` zu finden. Dies entspricht einer optimalen Anpassung des Language Models an den akustischen Teil des Spracherkenners.

Um die Anpassung beurteilen zu können, muß die Güte der Erkennung gemessen werden. Eine oft benutzte Möglichkeit hierzu ist die Bestimmung der Wortfehlerrate, die dadurch bestimmt wird, daß alle Abweichungen von Satzhypothese und Satzvorlage ins Verhältnis zu der Wortanzahl der Satzvorlage gesetzt werden. Als Abweichungen unterscheidet man dabei Ersetzungen (Substitutions), Einfügungen (Insertions) und Löschungen (Deletions) von Worten. Die Erkennungsleistung (Performance) ist die von 1 subrahierte Wortfehlerrate:

$$WA = 1 - \frac{N_{Substitutions} + N_{Insertions} + N_{Deletions}}{N_{All}}$$

Für das Alignment von Satzhypothese mit Satzvorlage wird ein Verfahren der dynamischen Programmierung verwendet. Um nun die optimalen `lmWeight` und `lmPenalty` zu finden, muß nur jeweils für jedes mögliche Paar von `lmWeight` und `lmPenalty` über alle Testäußerungen die Summe der Performance WA berechnet werden. Das Paar von `lmWeight` und `lmPenalty`, dessen Performancesumme am größten ist, ist dann auch mit hoher Wahrscheinlichkeit für ungesehene Äußerungen die optimale Besetzung, solange das Suchraster fein genug gewählt wurde und einen ausreichend großen Bereich überdeckt.

#### 5.5.6.2 Performance des kontextunabhängigen Erkenners

Nach den ersten Trainingsiterationen lag die Performance unerfreulich niedrig bei unter 10 %. Als Ursache dafür stellten sich aber einige knifflige Portierungsprobleme heraus, die anhand eines Vergleichserkenners unter Unix isoliert und schließlich auch eliminiert werden konnten. Der erste fehlerfreie Erkenner, der auf ca. 70 Sprechern trainiert wurde, erreichte dann eine Performance von 47,1 % (Beam 200/TopN 200) mit einem verhältnismäßig kleinem Language Model aus ca. 10 Mio. Worten. Diese Performance war enttäuschend, so daß vorerst kein kontextabhängiger Erkenner darauf aufbauend trainiert wurde, sondern sich mit der Suche nach den Ursachen des schlechten Abschneiden beschäftigt wurde. Dieser Vorgang war sehr langwierig, förderte aber dann die folgenden Probleme zutage, die teilweise gelöst werden konnten:

- Beim Vergleich der Orginaltexte mit den Audiodaten wurden manchmal Korrekturen vorgenommen, die die 2-Byte-Struktur zerstörten. Wenn in den chinesischen Text eine ungerade Anzahl von ASCII-Zeichen eingefügt wurde, wurden ab dieser Stelle nicht mehr die richtigen beiden Byte zu einem chinesischen Zeichen zusammengefaßt. Dies war deshalb problematisch, da der Editor eine Mischung aus ASCII und GB-Code darstellen konnte, und es dem Korrekturleser daher nicht aufgefallen ist. Der Pinyinkonverter aber erwartete reinen GB-Code zur Konvertierung. Die häufigsten Einfügungen waren einzelne Leerzeichen oder die ungültige Wandlung von GB-Code Zahlen in ASCII-Zahlen (z.B. 918 (ASCII) statt ９１８（GB）). Um diese Fehler aufzufinden und zu korrigieren wurde ein Suchskript in Perl implementiert.
- Ein weiteres Problem bestand darin, daß das Janustool, welches die Audioäußerungen anhand kurzer Pausen trennte, die Zuordnungsnummern der Audiodateien für Sprechernummern größer 100, teilweise falsch vergeben hat. Um diese Zuordnung zu korrigieren, wurde die Satzreihenfolge in den Orginaltexten an die Audiodateibezeichnungen angepaßt, da dies einfacher möglich war und auch eine einfache Synchronisation der Unix und Windows-Datenbestände ermöglichte.
- Dadurch daß keine frei verfügbaren chinesische Wörterbücher zur Zeit der Entwicklung des Pinyin-Tools existierten, wurde ein Wörterbuch halbautomatisch aus Texten und Wortlisten generiert. Dieser Prozeß war natürlich mit vielen Fehlern behaftet, da das entstandene Wörterbuch aufgrund seiner Größe von über 100000 Wörtern nicht mehr von Hand vollständig korrigiert werden konnte. Die daraus resultierende falsche Pinyinzuordnung beeinflußte in nicht unwesentlichem Maße auch die Erkennung. Der chinesische Spracherkenner war aber auch eine ideale Hilfe diese Fehler im Wörterbuch aufzufinden. Dazu konnten zwei Verfahren angewandt werden. Erstens konnte beim Schreiben der Labels der Wert des Beam-Parameters sehr klein eingestellt werden, so daß für Äußerungen mit falschen Pinyin-Zuordnungen keine Backtrace bei der Viterbizuordnung gefunden werden kann. Die Äußerungen ohne erzeugte Labeldatei waren somit Kandidaten für falsche Pinyin-Zuordnungen. Zweitens können auch die Äußerungen bei der Erkennung aussortiert werden, die unter einer gewissen Performance liegen, oder es können die Wortvertauschungslisten, die beim Vergleich der Hypothesen mit den Referenztexten erzeugt werden, analysiert werden. Mittels dieser Hilfen konnten einige häufige Umsetzungsfehler aufgespürt werden.
- Aufgrund der Tatsache, daß die Aufnahmen von Chinesen unterschiedlicher Herkunft und Bildung stammen, sind die Sprachdaten sehr unterschiedlich. Die Lesegeschwindigkeit variiert von langsam bis sehr schnell, wodurch unterschiedliche Koartikulationen auftreten.

Die Sprachdaten sind, im Gegensatz zu einem Diktiersystem, nicht kooperativ gelesen worden, d.h. die meisten Sprecher bemühten sich nicht, besonders deutlich zu sprechen.

- Bei einigen Sprechern wurde eine Übersteuerung des Aufnahmepegels festgestellt. Bei übersteuerten Äußerungen konnte dann auch ein starker Einbruch der Erkennungs-Performance ermittelt werden. Durch Normierung der Amplitude und anschließender leichter Tiefpaßfilterung konnte, bei einzelnen Äußerungen eine Verdopplung der Erkennungs-Performance bewirkt werden. Neben einer Übersteuerung machte auch eine wesentlich zu geringe Aussteuerung (unter 10 %) dem Erkenner Schwierigkeiten. Durch eine Anhebung konnte auch hier eine wesentliche Verbesserung der Performance der betroffenen Äußerung erreicht werden. Diese Ergebnisse überraschen auf Grund der schon vorhandenen Normierung während der Vorverarbeitung. Diese Normierung wurde nicht beim Training eingesetzt, ist aber sicher eine weitere Untersuchung wert!
- Englische Abkürzungen und Namen, sowie chinesische Eigennamen verursachen erwartungsgemäß größere Probleme. Das Pinyinsystem wurde dementsprechend erweitert, daß es auch englische Buchstaben auf Zeichen abzubilden vermochte.

Neben diesen Fehlerbereinigungen wurden noch folgende Verbesserungen vorgenommen:

- Da nun alle Trainingsdaten zur Korrektur gelesen waren, konnte nun der gesamte Trainingsdatenbestand verwendet werden. Statt ca. 70 Sprechern wurden nun alle 112 Sprecher der Trainingsmenge zum Training verwendet.
- Der Textkorpus des Language Models wurde von 10 Mio. auf 82 Mio. aufgestockt, und nun unter Unix mit dem etwas leistungsfähigeren Tool `ngrammodel` erzeugt, bevor es später wieder auf das Windowssystem zurück kopiert wurde.
- Dadurch, daß das System schon auf ein trainiertes aufsetzen konnte, sind die 4 verwendeten Iterationen „Labels schreiben bis Training“ nun effektiver.

Nach diesen Fehlerbereinigungen und Verbesserungen stieg die Performance des Systems dann auch auf folgende Werte (80 Äußerungen aus der Evaluationsmenge):

| Beam\TopN | 100 | 200 |
|---|---:|---:|
| 130 | 66.4 % | 66.7 % |
| 200 | 68.4 % | 69.2 % |

Dies ist eine Steigerung um mehr als 47.1 % auf 69.2%, so daß nun eine Basis für das Erzeugen eines kontextsensitiven Systems geschaffen worden ist.

## 5.6 Aufbau eines kontextabhängigen Erkenners

Bei einem kontextabhängigen Erkenner geht man von der Überlegung aus, daß der unmittelbare sprachliche Kontext eines Phonems dessen Aussprache beeinflußt. Allerdings können nicht für jeden Kontext Streamparameter trainiert werden, da schon bei einer Kontextbreite von 1 (jeweils ein linkes und ein rechtes Phonem) die ungeheure Anzahl von $|\text{Phonemanzahl}|^3$ mögliche Modelle zu trainieren wäre. Beim chinesischen Erkenner wären das über eine Million Modelle, so daß die Trainingsdaten bei weitem nicht ausreichen würden, um alle Modelle nur einigermaßen zu trainieren, ganz abgesehen vom enormen Zeit- und Speicheraufwand. Deshalb wurden die Kontexte zu Klassen zusammengefaßt. Dazu wurden zuerst mittels des

Janus-skripts `Ptree.tcl` alle in den Trainingsdaten vorhandenen Kontexte erfaßt und deren Auftreten gezählt. Modelle des gleichen Phonems teilen sich dann ein Codebuch während jedes Modell seine eigene Verteilung hat. Das Janusskripts `trainPT.tcl` trainiert dann diese Modelle, bevor hernach mit dem Janusskript `ClusterCB.tcl` ein Clusterbaum aufgebaut wird, der dann durch das Janusskript `SplitCB.tcl` in eine bestimmbare Anzahl von Klassen mit den ihnen zugeteilten Modellen, geteilt wird. Dabei gibt es die beiden grundsätzlich verschiedenen Vorgehensweisen:

1. Agglomerative Ballung (botton-up)
   Es werden immer die zwei ähnlichsten Klassen vereinigt, bis die vorgegebene Anzahl von Klassen erreicht wurde. Eventuell müssen während der Klassenvereinigung einige Modelle in andere Klassen verschoben werden, um eine optimale Clusterung zu erhalten. Bei diesem Verfahren ist keinerlei zusätzliches sprachliches Wissen notwendig. Es werden aber auch nur die schon bekannten Elemente berücksichtigt, so daß neue Elemente nicht behandelt werden können.
2. Divisive Ballung (top-down)
   Es wird ausgehend von einer Klasse mit allen Modellen, anhand von linguistischen Fragen, immer die Gruppe, die den höchsten Informationsgewinn verspricht, geteilt. Das Verfahren ist beendet, sobald die gewünschte Anzahl von Klassen erreicht wurde.
   Bei diesem Verfahren wird externes linguistisches Wissen benötigt und im allgemeinen auch keine optimale Clusterung erreicht. Andererseits ist dieses Verfahren sehr effizient und es können auch ungesehene Elemente berücksichtigt werden.

Das Janusskripts `ClusterCB.tcl` benutzt das zweite Verfahren. Als linguistische Fragen wird auf die Zugehörigkeit zu einer Phonemklasse gefragt.

Die im Falle des chinesischen Erkenners definierten linguistischen Fragen umfassen:

> SILENCE, NOISES, CONSONANT, BILABIAL, ALVEOLAR, VELAR, AFFRICATE, PLOSIVE, NASAL, FRICATIVE, VOICED, UNVOICED, TRIPHTHONG, VOWEL, DIPHTHONG, TRIPHTHONG, Phoneme mit bestimmtem Ton, bestimmte Teilmengen von Vokalkonstrukten.

Allerdings haben Untersuchungen gezeigt, daß die Anzahl und die Definition der Phonemklassen keinen zu großen Einfluß auf die Güte der Clusterung ausübt.

Für den aufgebauten chinesischen Erkenner wurde eine Kontextbreite von 1 und eine Anzahl von 3000 geclusterten Modellen gewählt. Durch Vergrößern dieser Werte kann die Anpassung an den Kontext sicher noch vergrößert werden, wovon hier aber aufgrund von Rechenzeit- und Hauptspeicherproblemen abgesehen wurde.

Zusätzlich zur Betrachtung des Kontextes wurde jetzt auch das VTLN-Verfahren zur Sprechernormalisierung angewandt. Bei diesem Verfahren werden die Äußerungen normiert, indem beim Training für jeden Sprecher zuerst ein eigener Normierungsfaktor anhand einiger stimmhafter Sprachproben bestimmt wird und dann alle Trainingsdaten dieses Sprechers mit den so transformierten Sprachdaten trainiert werden. Beim Test werden zuerst alle möglichen Normierungsfaktoren ausprobiert, um dann den, der den besten Score liefert, auszuwählen. Man hofft durch das Verfahren die Varianz innerhalb der Klassen zu minimieren und die Trennbarkeit zu erhöhen.

Nach zwei weiteren Iterationen von „Label schreiben“ bis „Training“, wurden dann von diesem kontextabhängigen System folgende Performancewerte erreicht:

| | Ohne VTLN | Mit VTLN |
|---|---|---|
| Beam = 200, TopN = 200 | 76,3 % | **77,1 %** |

Es wurden die gleichen 80 Äußerungen aus der Evaluationsmenge verwendet, wie bei dem kontextunabhängigen System. Trainingsmenge und Language Model blieben unverändert.

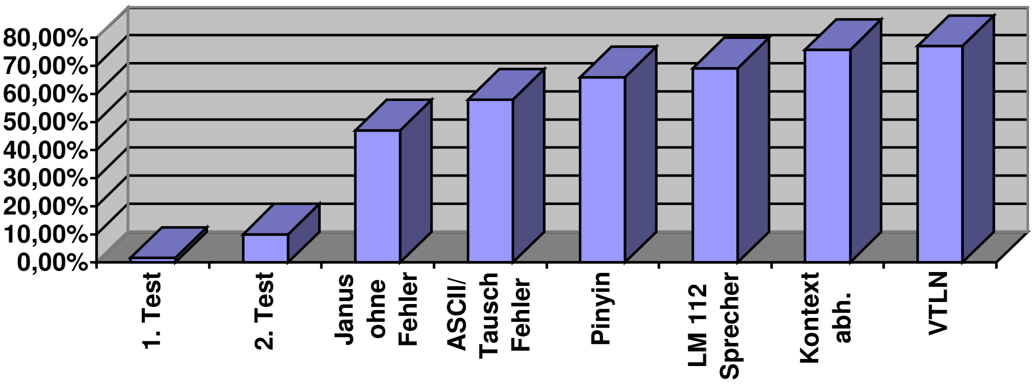


Abbildung 5.19: Performanceentwicklung

Obige Abbildung gibt die Fortschritte in der Erkennungleistung wieder. Die mit “1.Test” und “2.Test” bezeichneten Systeme sind diejenigen, deren Leistung aufgrund Portierungsfehler noch stark beeinträchtigt war. Das erste fehlerfreie System erreichte 47,1 %. Die Bereinigung der ASCII-Einstreuung, sowie die Korrektur der falschen Audiozuordnungen und andere kleinere Fehler, ließen die Performance auf über 57 % steigen. Verbesserungen an der Pinyinumsetzung ergaben eine weitere Leistungssteigerung um 8-9 Prozentpunkte. Schließlich wurde das Languagemodel aus dem gesamten Textkorpus aus 82,5 Mio Worten berechnet, und alle 112 Trainingssprecher zum Training verwendet. Das System erreichte damit 69,2 % Erkennungsleistung. Der Aufbau eines kontextunabhängigen Systems aus 3000 Polyphonmodellen steigerte die Erkennungsrate auf 75,9 %. Mit zusätzlicher Sprechernormierung erreichte der Spracherkenner die oben aufgeführten 77,1 %.

## 5.7 Aufbau eines Silbenerkenners

Da es im Chinesischen nur eine verhältnismäßig kleine Anzahl von Silben gibt, lag es nahe, auch den Ansatz der Silbenerkennung zu untersuchen. Beim Aufbau des Silbenerkenners wurden alle Konfigurationdateien vom vorherigen chinesischen Erkenner übernommen. Nur das PhonesSet und das Aussprachewörterbuch mußten angepaßt werde. Dies war Dank der parameterisierbaren Erzeugungsskripte schnell geschehen. Auf eine Erhöhung der Subphonemanzahl von derzeit 3, wurde, obwohl dies wegen der nun wesentlich längeren Phoneme (Silben) sicher sinnvoll wäre, zuerst einmal verzichtet.

Insgesamt wurden 1269 Phoneme erzeugt, wobei diese jetzt genau den bereinigten Pinyinaussprachesilben der chinesischen Sprache entsprechen. Durch die hohe Phonemanzahl war nun eine manuelle Erstellung der Mapping-Tabelle, zur Initialisierung des Erkenners anhand des leistungsfähigen existierenden Erkenners, zu arbeitsintensiv. Sie wurde nun durch das erstellte Perlskript `Make_Mapping` automatisch erzeugt.

Nach 3 Iterationen (Labels schreiben bis Training) konnte dann folgende Performance des kontextunabhängigen Silbenerkenners erreicht werden (gleiche Testbedingungen wie das vorherige kontextunabhängige System):

| | Performance |
|---|---:|
| Beam = 200, TopN = 200 | **70,9 %** |

Dies ist ein Performancegewinn von 1,7 % gegenüber dem kontextunabhängigen Phonemerkenner. Allerdings steigt der Speicherplatzbedarf für die Codebücher um den Faktor 10 und der Zeitbedarf für den Aufbau der Suchstruktur um den Faktor 60 auf über 17 Stunden. Janus scheint nicht so sehr effektiv mit Phonemanzahlen über 1000 umgehen zu können. Dies zeigte sich dann auch besonders, als ein kontextabhängiges System darauf aufbauend erzeugt werden sollte. Das verwendete Rechnersystem kam an seine Speichergrenzen und der zumutbare Rechenzeitbedarf wurde überschritten, so daß die Weiterentwicklung des kontextabhängigen Silbenerkenners abgebrochen werden mußte. Der Performancegewinn würde auch mit Sicherheit nicht so groß wie beim Phonemerkenner ausfallen, da durch die Silben ja schon breitere Kontexte erfaßt wurden.
Um sehr große Phonemanzahlen benutzten zu können, müßten Änderungen im Kern von Janus vorgenommen werden.

## 5.8 Erzeugung von chinesischen Zeichen aus der Pinyinumschrift

Mit dem Phonem- und Silbenerkenner stehen 2 leistungsfähige chinesische Spracherkenner zur Verfügung, die eine sprachliche Äußerung erfassen und als Ausgabe die wahrscheinlichste Hypothese für den gesprochenen Satz in der Pinyinumschrift zurückliefern. Nun ist aber die Pin-yinumschrift nicht das Resultat, welches ein Benutzer von einer chinesischen Spracherkennungslösung erwartet. Er möchte viel lieber eine Ausgabe in der für ihn gewohnten chinesischen Zeichenschrift.

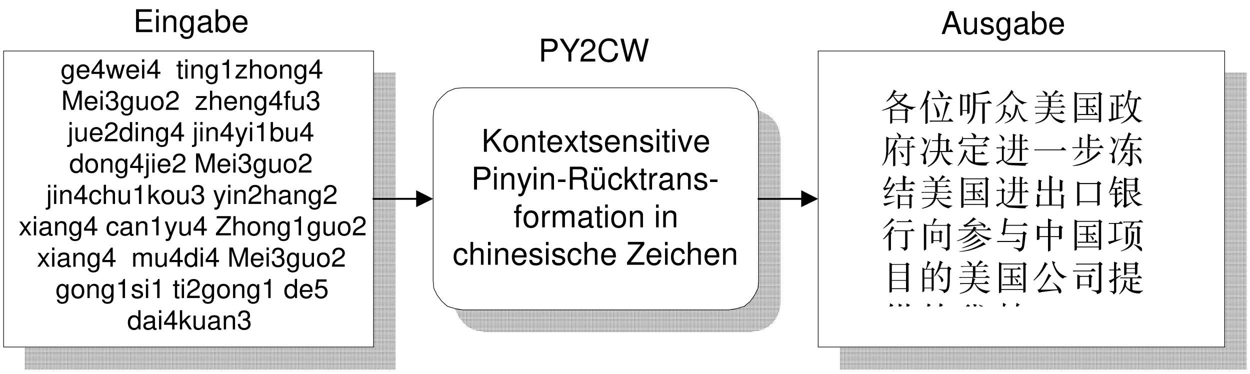

Eingabe
ge4wei4 ting1zhong4
Mei3guo2 zheng4fu3
jue2ding4 jin4yi1bu4
dong4jie2 Mei3guo2
jin4chu1kou3 yin2hang2
xiang4 can1yu4 Zhong1guo2
xiang4 mu4di4 Mei3guo2
gong1si1 ti2gong1 de5
dai4kuan3
PY2CW
Kontextsensitive Pinyin-Rücktransformation in chinesische Zeichen
Ausgabe
各位听众美国政
府决定进一步冻
结美国进出口银
行向参与中国项
目的美国公司提

Abbildung 5.20: Rücktransformation der Pinyinumschrift in chinesische Zeichen

Wie schon im 2. Kapitel erläutert, ist eine Rücktransformation der Pinyinumschrift in chinesische Zeichen mehrdeutig. Im Durchschnitt kommen auf jede Pinyinaussprachesilbe mehr als 10 potentielle Kandidaten alleine aus den gebräuchlichsten chinesischen Zeichen. Diese enorme Mehrdeutigkeit kann nur anhand des Kontextes aufgelöst werde. Und auch das ist nicht immer möglich. So kann z.B. eine chinesische Sekretärin beim Diktat, ihr unbekannte Eigennamen und Fremdwörter nicht niederschreiben, ohne sich vorher explizit über die Schreibweise erkundigt zu haben. Im Deutschen oder Englischen kann im Gegensatz dazu in den meisten Fällen, anhand gelernter Rechtschreibregeln, die korrekte Schreibweise niedergeschrieben werden.

In dieser Diplomarbeit wurde ein Verfahren entwickelt, welches für einen Satz in Pinyinumschrift eine entsprechende Satzhypothese aus chinesischen Zeichen, ausschließlich aus Kontext-informationen erzeugt. Dabei ist die Wortrennung, die der chinesische Erkenner zurückliefert eine große Hilfe.
Das implementierte Verfahren erlernt anhand von Beispieltexten, die sowohl in Pinyinumschrift, als auch in Zeichenschrift vorliegen, Umsetzungsregeln. Da das Pinyinprogramm beliebig viele Texte des Textkorpus umsetzen kann, existiert in dieser Konvertierungsrichtung nicht das Problem der „spärlichen Daten“. Genügend Beispieldaten zu haben ist eine wichtige Voraussetzung, um überhaupt verläßliche Schätzungen vornehmen zu können.

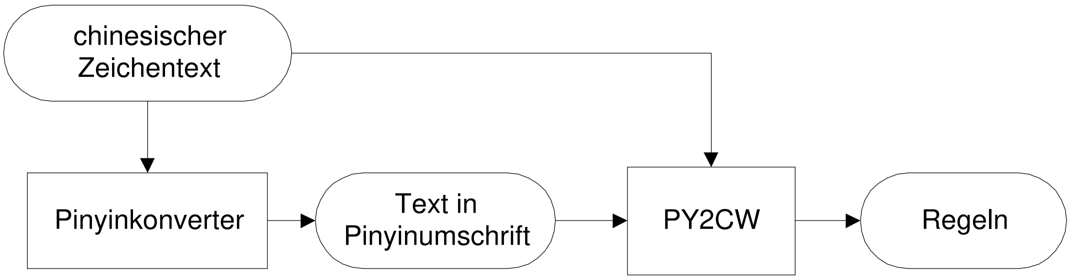


Abbildung 5.21: Erzeugung der Umsetzungsregeln

Die Umsetzungsregeln werden durch das Perlskript `PY2CW` erzeugt. Jede Regel besteht im Bedingungsteil aus einer Pinyinphrase und im Ersetzungsteil aus der entsprechenden Zeichenphrase. Im Umsetzer wird dann immer die Regel mit dem spezifischsten d.h. längsten Bedingungsteil gewählt.
Ziel der Regelerzeugung ist, daß die Umsetzung eine hohe Performance liefert, mit der Nebenbedingung, die Regelbasis zu minimieren. Dazu wird der Textkorpus in mehreren Pässen durchlaufen. Beginnend mit der Kontextbreite 0 (=Phrasenlänge 1) wird für jede Phrase die wahrscheinlichste Umsetzung durch Zählen aller vorkommenden Umsetzungen ermittelt. Diese wahrscheinlichsten Umsetzungen werden dann in die Regelbasis aufgenommen.

Beispiel für die Phrase „shi4“:

```
shi4 -> 是   5637284
shi4 -> 市   2114836
shi4 -> 事   1613194
shi4 -> 试   380592
shi4 -> 似   143499
shi4 -> 世   865036
shi4 -> 室   246235
shi4 -> 式   897246
shi4 -> 饰   69100
shi4 -> 誓   24239
shi4 -> 士   484345
shi4 -> 适   359341
....
```

`shi4 -> 是` ist die wahrscheinlichste Umsetzung und wird somit der Regelbasis hinzugefügt.

Im nächsten Pass durch den Textkorpus, nun mit Kontextbreite 1, wird die Umsetzung der Pinyinwörter anhand der zur Verfügung stehenden Regeln vorgenommen. Jede Umsetzung wird anhand des Orginaltextes kontrolliert. Falls ein Umsetzungsfehler auftritt, werden Ausnahmen mit der aktuellen Kontextbreite generiert. Von den sich teilweise widersprechenden Ausnahmen werden wiederum die wahrscheinlichsten in die Regelbasis aufgenommen.

Beispiel:

```
 由   市    种子        => Ausnahme1: you2 shi4 -> 由 市
you2 shi4 zhong3zi3    => Ausnahme2: shi4 zhong3zi3 -> 市 种子

 由   事    种子        => Ausnahme1: you2 shi4 -> 由 事
you2 shi4 zhong3zi3    => Ausnahme2: shi4 zhong3zi3 -> 事 种子
```

```
you2 shi4 -> 由 事  3491
you2 shi4 -> 由 市   141
....
```

`you2 shi4 -> 由 事` ist die wahrscheinlichste Umsetzung und wird somit der Regelbasis hinzugefügt.

In den darauffolgenden Pässen wird mit wachsender Kontextbreite auf dieselbe Weise verfahren.

Das Umsetzungsverfahren wurde aufbauend auf diese grundsätzliche Vorgehensweise weiter verfeinert:

- Es wurde ein Mindestvorkommen von Ausnahmeregeln im Textkorpus festgelegt, so daß nicht eine seltene Situation eines häufig vorkommenden Kontextes die guten Regeln einer geringeren Kontextbreite aushebeln kann.
  In einem sehr breiten Kontext können auch zwei verschiedene Umsetzungen eines Wortes formal korrekt sein. Die richtige läßt sich dann nur über Metawissen ermitteln. Ohne das Festlegens eines Mindestvorkommens würden in einer solchen Situation unnötige Regeln erzeugt.
  Auch läßt sich durch das Festlegens eines Mindestvorkommens die Größe der Regelbasis effektiv beschränken.
- Für Situationen in denen sich die richtige Umsetzung nur über Metawissen feststellen läßt, oder mehrere Umsetzung gleichbedeutend sind, wurde eine Synonymliste erstellt. Somit

werden nur Ausnahmen erzeugt, wenn ein Umsetzungswiderspruch zu allen Synonymen besteht. Zu solchen Synonymen zählen z.B. Zahlen verschiedener chinesischer Zahlensysteme, der unterschiedlich angewandte „de“-Partikel und unterschiedlich geschriebene aber gleich ausgesprochene Geschlechtsformen.

- Die Signifikanz einer erzeugten Ausnahme wurde unterschiedlich bewertet, je nachdem an welcher Position der Phrase der Umsetzungsfehler aufgetreten ist.

Der Textkorpus, der zum Training verwendet wurde, umfaßt 4 Jahrgänge der chinesischen Tageszeitung People’s Daily. Er ist eine Teilmenge (ca. 50%) des Textkorpus zu Erzeugung des Laguagesmodels. Die Regelbasis besteht aus ca. 530000 Regeln.

Das Umsetzungstool kann satzweise die Pinyinumschrift, den Orginalsatz und die Satzhypothese untereinander ausgeben und dazu eine Fehlerstatistik ausgeben:

Beispiel:

```
;SprecherID 080
Orginal:   ;SprecherID 080
Hypothese:
; 1:
ke1xie2 wu3 da4 dai4biao3 he2 te4 yao1 lie4xi2 dai4biao3 yi1qian1 duo1 ren2 chu1xi2 le5 bao4gao4
hui4
Orginal:   科协 五 大 代表 和 特 邀 列席 代表 1 0 0 0 多 人 出席 了 报告 会
Hypothese: 科协 （武 大 -> 五 大） 代表 和 特 邀 列席 代表 1 0 0 0 多 人 出席 了 报告 会
; 2:
wo3guo2 zhao1zuo4quan2 fa3 shi2shi1 cheng2xiao4 zhuo1zhu4
Orginal:   我国 著作权 法 实施 成效 卓著
Hypothese: 我国 著作权 法 实施 成效 卓著
; 3:
xin1hua2she4 ji4zhe3 qu3 zhi4 hong1
Orginal:   新华社 记者 曲 志 红
Hypothese: 新华社 记者 曲 志 红
; 4:
yu2 zhong1guo2 de5 da4duo1shu4 gong1min2 lai2shuo1 yi3 jue2fei1 shi4 ge4 mo4sheng1 de5 gai4nian4
le5
Orginal:   于 中国 的 大多数 公民 来说 已 绝非 是 个 陌生 的 概念 了
Hypothese: （<s> 虞 -> <s> 于） 中国 的 大多数 公民 来说 （以 -> 已） 绝非 （市 各 -> 是 个） 陌生 的
概念 了
; 5:
Orginal:   随着 中华人民共和国 著作权 法 的 实施 5 年来 版权 保护 越来越 引起 人们 的 高度 关注
Hypothese: 随着 中华人民共和国 著作权 法 的 实施 5 年来 版权 保护 越来越 引起 人们 的 高度 关注

....

Korrekt: 1406
Fehler: 83  (5.57% Wortbezogen, 2.97% Zeichenbezogen)
  Tausch: 83
  OOV: 0
```

Diese Darstellung erlaubt eine weitere Analyse von Umsetzungsfehlern. Es zeigte sich, daß wie erwartet, die meisten Fehler bei Eigennamen, sowie bei englischen Abkürzungen auftraten. Eine gründliche Überarbeitung und Anpassung des Vokabulars könnte somit die Fehlerrate weiter stark senken.

## 5.9 Gesamtperformance des chinesischen Erkenners

Wenn nun der Erkennungsprozeß mit der Pinyinumsetzung kombiniert wird, gehen beide Fehler in die Gesamtperformance ein.

Die Performancedaten waren bis jetzt immer auf die Anzahl der Worte bezogen. Dies erleichtert einen Vergleich zu Spracherkennern westlicher Sprachen, die die Fehlerrate bzw. die Performance (Wortakkuratheit) immer auf die Anzahl der Worte beziehen. Um den chinesischen Spracherkenner aber mit anderen chinesischen Erkennern zu vergleichen, ist die Fehlerrate pro Zeichen ausschlaggebend, da die Wortdefinition im Chinesischen nicht eindeutig ist. Die auf Zeichen bezogene Performance ist immer besser als die auf ganze Worte bezogene, da bei einem falsch erkannten Wort oft noch Wortteile (Aussprachesilben/Zeichen) korrekt sind. Einen Einfluß hat auch, ob bei der Aufaddierung der Fehler diese mit der Satzlänge gewichtet wurden oder diese unabhängig von der Satzlänge einfach aufaddiert wurden. Im letzten Fall nimmt die Fehlerrate etwas zu, da kurze Sätze anfälliger für Fehler sind.

Im folgenden sind alle Performancedaten, sowohl für den Phonemerkenner, als auch für den Silbenerkenner nach diesen Gesichtspunkten nochmals zusammengefaßt. Jede Angabe wurde aus denselben 70 Testäußerungen der Evaluationsmenge berechnet.

| | | **Satzlänge wurde gewichtet** | **Satzlänge wurde nicht gewichtet** |
|---|---|---|---|
| **Pinyinausgabe** | **Auf Worte bezogen** | 77.3% | 75.9% |
| | **Auf Zeichen bezogen** | 85.5% | 84.4% |
| **Zeichenausgabe** | **Auf Worte bezogen** | 74.7% | 73.3% |
| | **Auf Zeichen bezogen** | **83.0%** | 81.3% |

Abbildung 5.22: Performancedaten des phonembasierten chinesischen Spracherkenners

| | | **Satzlänge wurde gewichtet** | **Satzlänge wurde nicht gewichtet** |
|---|---|---|---|
| **Pinyinausgabe** | **Auf Worte bezogen** | 73.3% | 70.5% |
| | **Auf Zeichen bezogen** | 82.3% | 78.7% |
| **Zeichenausgabe** | **Auf Worte bezogen** | 69.8% | 66.9% |
| | **Auf Zeichen bezogen** | **79.7%** | 76.1% |

Abbildung 5.23: Performancedaten des silbenbasierten chinesischen Spracherkenners

In der linken Spalte wird die Wortakkuratheit aufgeführt, während in der rechten Spalte der Mittelwert der einzelnen Performacewerte der Sätze aufgeführt ist. Der zweite Wert kann für Infomationssysteme von Bedeutung sein, in denen die Anfragen bewertet werden. Die Pinyinausgabe bezieht sich auf die wortgetrennte Lautschriftausgabe. Bei der Zeichenausgabe wird die Pinyinausgabe in Zeichen konvertiert, bevor die Performance gemessen wird. Die Performanceangaben sind sowohl auf Worte als auch auf Zeichen bezogen angegeben. Die erste Angabe eignet sich zum Vergleich mit westlichen Sprachen, während die zweite sich zum Vergleich mit zeichenbasierten Sprachen eignet.

# 6 Zusammenfassung und Ausblick

Die sprecherunabhängige Spracherkennung mit großem Vokabular ist gegenwärtig ein wichtiges Forschungsgebiet, welches vielerlei Anwendungsgebiete, wie Auskunftssysteme, Diktierlösungen, Bedienung elektrischer Geräte, Multimedia, Call- und Servicecenter, Übersetzungssysteme und dergleichen erobert. Es wurden in den letzen Jahren beachtliche Fortschritte erzielt, um die wichtigsten Spracherkennungsprobleme, nämlich die Erkennung kontinuierlicher, spontaner Sprache auf einem sehr großen Vokabular trotz der vielfältigen akustischen Variabilität, zu lösen.

Die „Spracherkennung im Chinesischen“, mit der sich die vorliegende Arbeit beschäftigt, ist von besonderer Bedeutung, da im Chinesischen die Texteingabe mittels einer Tastatur einen unvergleichbar höheren Aufwand als in westlichen Ländern darstellt, und somit andere Eingabemethoden auf eine hohe Akzeptanz stoßen. Die Spracherkennung, als natürlichstes Kommuniktionsmedium, scheint hier geradezu prädestiniert zu sein, das neue Standardeingabeverfahren zu werden. Da die chinesische Sprache in vielerlei Hinsicht von den uns geläufigen Sprachsystemen verschieden ist, müssen deren Besonderheiten und Konzepte speziell betrachtet werden. Als die wichtigsten außergewöhnlichen Eigenschaften stellten sich das Fehlen eines definierten Wortmodells, die bedeutungstragende Tonalität und die fehlende direkte Bindung der Aussprache an die Schriftsprache heraus.

Unter Berücksichtigung der besonderen Charakteristiken der chinesischen Sprache wurden Ansätze zum Aufbau eines Spracherkenners untersucht und im Laufe der Arbeit ein auf Phonemen basierender und ein auf Silben basierender chinesischer Spracherkenner entwickelt. Dabei stützen sich die Untersuchungen und Entwicklung auf den Spracherkennungsteil des Sprach-zu-Sprach-Übersetzungssystems Janus, welches während dieser Arbeit von Unix auf das Betriebssystem Windows portiert und um eine Anzahl von Hilfsprogrammen erweitert wurde. Die entwickelten Spracherkenner erreichen sprecherunabhängig eine auf Zeichen bezogene Performance von 83.0% für den kontextabhängigen Phonemerkenner bzw. 79.7% für den kontextunabhängigen Silbenerkenner. Die Ausgabe besteht aus einer Satzhypothese in chinesischen Zeichen. Die Spracherkenner sind sowohl unter verschiedenen Unixdialekten, als auch unter Windows einsetzbar.

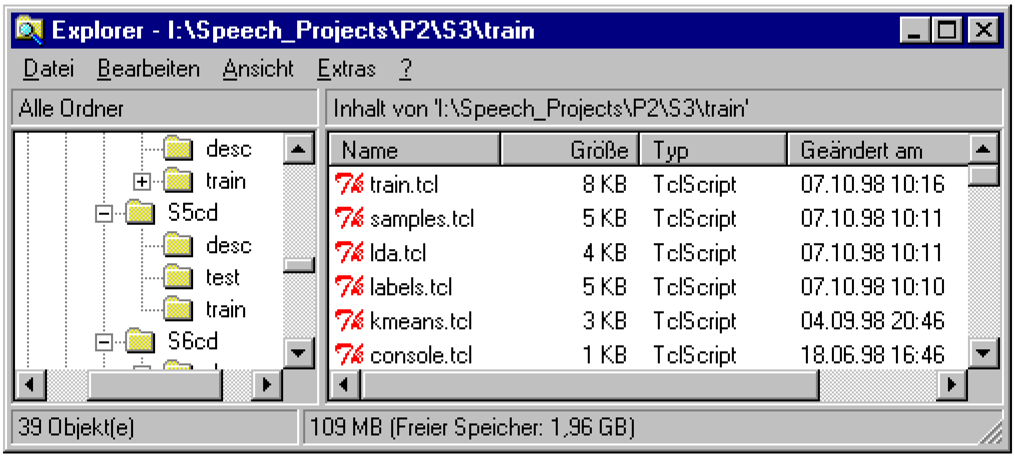
Ein weiterer Schwerpunkt dieser Arbeit war die Entwicklung einer grafischen Benutzerschnittstelle für das Janussystem. Deren Aufgabe ist es, den Einarbeitungsaufwand eines neuen Benutzers zu reduzieren, indem dieser gezielt durch den Aufbau eines Spracherkenners geführt wird. Die in die Entwicklungsumgebung integrierten Werkzeuge sollen den Benutzer von Routinearbeit entlasten und die Fehleranalyse vereinfachen. Das Ziel, welches hinter dieser Entwicklung stand war, daß man sich während wissenschaftlichen Untersuchungen auf die Hauptpunkte konzentrieren kann, ohne durch Details abgelenkt zu werden.

Weitere interessante Ansätze, die in dieser Arbeit nicht verfolgt wurden, wären eine zusätzliche explizite Erfassung der Pitchinformation, den Einsatz von neuronalen oder hybriden Systemen, die aufgrund des geringen Silbenumfangs aussichtsreich sein könnten und die Berücksichtigung von Aussprachevarianten.

# Anhang A: Verschiedenes

In diesem Teil soll noch auf einige weitere Tools und Besonderheiten des Janussystems unter Windows hingewiesen werden:

- In Janus wurde der Befehl `Beep` integriert. Dieser Befehl ermöglicht z.B. eine Signalisierung, wenn ein Skript beendet wurde oder eine Ausnahmesituation auftritt. Der `Beep`-Befehl erzeugt ein ca. eine Sekunde langes Signal aus dem Computerlautsprecher, ohne eine Soundkarte zu benötigen.
- Janusskripte können auch direkt aus dem Explorer durch Doppelklick gestartet werden, solange alle Parameter innerhalb des Skripts belegt wurden.


Explorer - I:\Speech_Projects\P2\S3\train
Datei
Bearbeiten
Ansicht
Extras
?
Alle Ordner
Inhalt von 'I:\Speech_Projects\P2\S3\train'
desc
train
S5cd
desc
test
train
S6cd

| Name | Größe | Typ | Geändert am |
|---|---|---|---|
| train.tcl | 8 KB | TclScript | 07.10.98 10:16 |
| samples.tcl | 5 KB | TclScript | 07.10.98 10:11 |
| lda.tcl | 4 KB | TclScript | 07.10.98 10:11 |
| labels.tcl | 5 KB | TclScript | 07.10.98 10:10 |
| kmeans.tcl | 3 KB | TclScript | 04.09.98 20:46 |
| console.tcl | 1 KB | TclScript | 18.06.98 16:46 |

39 Objekt(e)

109 MB (Freier Speicher: 1,96 GB)

Abbildung A.1: Windows Explorer

- Um die chinesische Grammatik von wortwörtlich aus dem Deutschen/Englischen übersetzten Phrasen zu verbessern, wurde das Skript `LMScore` erstellt, daß jeweils einen Wahrscheinlichkeitwert für verschiedene Wortstellungen anhand des chinesischen Languagemodels (Trigramme mit Rückfall auf Bi-/Unigramme) zurückliefert. Man hofft, daß die wahrscheinlichste Wortstellung auch die grammatikalisch richtige ist.
  Damit die Berechnung auch bei mehreren 100 MB großen Languagemodellen sehr schnell geht, wird, sofern kein Suchindex gefunden wird, ein solcher aufgebaut.
- Ein weiteres Experimentierfeld war die Sprachsynthese. Es wurde damit begonnen, aus anhand vom Spracherkenner gelabelter Sprachdaten, Sprachmuster zu extrahieren. Die Sprachmuster wurden dann entsprechend der Textvorlage zusammengesetzt und als Audiodaten ausgegeben.
- Da zum Training und Test von Spracherkennern ein sehr hoher Bedarf an Rechenzeit besteht, wurde nach Mittel und Wegen zur Auslagerung von Prozessen gesucht. Als Lösungsmöglichkeit, neben der Anschaffung neuer Hardware, fand sich der Einsatz eines X-Clients unter Windows, so daß Janusprozesse auf freien Unixrechnern ausgeführt werden konnten. Dabei stellte sich heraus, daß das Suchen nach freier Rechenkapazität und die daraus resultierende Prozeßzuteilung zu aufwendig war, um effektiv genutzt werden zu können. Dies war der Anlaß einen Prozeßscheduler zu entwickeln, der automatisch alle Rechner auf freie Rechenkapazität untersucht, und dann entsprechend die Prozesse verteilt. Dieser Scheduler läßt sich auch optimal über eine Telefoneinwahlleitung benutzen, da nach der Prozeßverteilung sofort die Verbindung wieder abgebaut werden kann. Eine

spezielle Funktion erlaubt, den Verarbeitungsstatus der Janusprozesse abzufragen. Der Scheduler wurde wiederum in Perl unter zu Hilfenahme des NET-Modules realisiert.

Scheduling-Logbuch:

```
i13a11.ira.uka.de:  2.27   (4 CPUs) => 2 Prozesse
nohup janusA trainVTLN.tcl >& log1 &
nohup janusA trainVTLN.tcl >& log2 &
i13a12.ira.uka.de:  0.00   (4 CPUs) => 4 Prozesse
nohup janusA trainVTLN.tcl >& log3 &
nohup janusA trainVTLN.tcl >& log4 &
nohup janusA trainVTLN.tcl >& log5 &
nohup janusA trainVTLN.tcl >& log6 &
i13s2.ira.uka.de:  0.36   (2 CPUs) => 1 Prozesse
nice nohup janusS trainVTLN.tcl >& log7 &
i13s7.ira.uka.de:  0.41   (2 CPUs) => 1 Prozesse
nice nohup janusS trainVTLN.tcl >& log8 &
```

Statusinformation:

```
Sprecher(Phonem) 74 von 112 bearbeitet (66,1%)
```

# Anhang B: Objekte in Janus

```
***********************************************************************
AModel

DESCRIPTION
acoustic model

METHODS
puts                   displays the contents of an amodel

***********************************************************************
AModelSet

DESCRIPTION
set of acoustic models

METHODS
puts                   displays the contents of an amodel set
add                    add a state graph to a set
get                    find acoustic model given a phonetic context
reset                  remove all amodels from the set

***********************************************************************
Cbcfg

DESCRIPTION
configuration of a codebook

METHODS
puts

***********************************************************************
Codebook

DESCRIPTION
Codebook

METHODS
puts
:=                     copy the parameters of one codebook into another
set                    set reference vectors in the codebook
cat                    cat one codecook behind another
createAccu             create an accumulator
freeAccu               remove an accumulator
createMap              create a codebook map
freeMap                remove a codebook map
splitList              codebook split candidates
split                  split codebook (create map)
plot                   plot a codebook into a canvas
covarTie               tie covariance matrices together
covarTie?              show which covariance matrices are tied together
covarUntie             untie covariance matrices
covarType              modify the type of covariance matrix
covarShift             add a constant value to all variances

***********************************************************************
CodebookAccu

DESCRIPTION
a single codebook's accumulator


METHODS
puts
:=                     copies one accumulator into another
+=                     adds one accumulator to another
*=                     multiplies an accumulator with a factor
set                    set reference vectors in the accumulator
clear                  reset a single codebook's accumulator to zero
subspace               define the accumulator subspacing

***********************************************************************
CodebookMap

DESCRIPTION
CodebookMap


METHODS
puts
add                    add items to the map
clear                  remove all items from the map

***********************************************************************
CodebookMapItem

DESCRIPTION
CodebookMapItem


METHODS
puts

***********************************************************************
CodebookSet

DESCRIPTION
Set of codebooks


METHODS
puts
index                  returns indices of named codebooks
name                   returns names of indexed codebooks
```

```
add                add a new codebook to the set
delete             remove codebook from the set
read               read codebook definitions from file
write              write codebook definitions to file
load               load codebook weights
save               save codebook weights
set                set and propagate defaultTopN or defaultRdimN
createAccus        creates accumulators for all codebooks
freeAccus          removes accumulators of  all codebooks
clearAccus         clear accumulators for all codebooks
loadAccus          loads codebook accumulators from a file
saveAccus          saves codebook accumulators into a file
createMaps         creates maps for all codebooks
freeMaps           removes maps of  all codebooks
split              split all codebooks
map                map all codebooks to new codebooks
update             update all codebooks

***********************************************************************
DBase

DESCRIPTION
DBase


METHODS
puts
open               open database
close              close database
add                add record to database
delete             delete record from database
read               add records from file to database
write              write records from database to file
get                get record from database
first              get first key in database
next               get next key in database
list               list all keys in database

TCL-DEFINED METHODS
uttInfo            find utterance information (dbaseUttInfo1)
***********************************************************************
DBaseIdx

DESCRIPTION
DBase Index Object


METHODS
puts
open               open index file
close              close index database
add                add record to index
delete             delete record from index
get                get record from index
first              get first key in index file
next               get next key in index file
list               list all keys in index file

***********************************************************************
DCovMatrix

DESCRIPTION
Covariance matrix type (double)

METHODS
puts
clear              clear the contents of an covariance accumulator
:=                 copy covariance matrix
+=                 add two covariance accumulators
*=                 scale the covariance accumulator

***********************************************************************
DMatrix

DESCRIPTION
Matrix of double values


METHODS
puts               print matrix contents as TCL list
resize             resize matrix
clear              set all matrix values to 0
copy               copy matrix
:=                 assign matrix (equiv. to 'copy')
get                get a single entry from a matrix
set                set a single entry in a matrix
FMatrix            convert from a FMatrix
trans              transpose matrix
unity              make matrix a unity matrix
mul                matrixA * matrixB
mulot              matrixA * matrixB'
eigen              eigenvalues and vectors of symmetric matrix
svd                singular value decomposition
inv                inverse of matrix using svd
simdiag            simultaneous diagonalisation

***********************************************************************
DVector

DESCRIPTION
Vector of double values


METHODS
puts               print vector as TCL list
resize             resize vector
copy               copy vector
:=                 assign vector (equiv. to 'copy')

***********************************************************************
```

```
DictWord

DESCRIPTION
Word with tagged phone transcription


METHODS
puts

**********************************************************************
Dictionary

DESCRIPTION
Set of words


METHODS
puts                display the contents of a dictionary
add                 add a new word to the set
delete              remove word from the set
read                reads a dictionary file
write               writes a dictionary file
index               return the internal index of a word
name                return the spelled word given the index

**********************************************************************
Distrib

DESCRIPTION
A single distribution


METHODS
puts
:=                  copies distribution weights
cat                      cat  one  distribution  behind  another  and  normalize
weights
createAccu          create a single distribution's accumulator
freeAccu            remove a single distribution's accumulator

**********************************************************************
DistribAccu

DESCRIPTION
a single distribution's accumulator


METHODS
puts
:=                  copies one accumulator into another
+=                  adds one accumulator to another
*=                  multiplies an accumulator with a factor
>=                  increase an accumulator's counts by a number
clear               reset a single distribution's accumulator to zero
```

```
**********************************************************************
DistribSet

DESCRIPTION
Set of distributions


METHODS
puts
index               returns indices of named distributions
name                returns names of indexed distributions
add                 add a new distribution to the set
delete              remove distribution from the set
score               computes the score of a mixture distribution
scoreNBest          computes the n-best mixtures mixtures
accuPath            accumulates sufficient statistic from path
accuFrame           accumulates sufficient statistic from frame
update              update distributions and codebooks
read                reads a distribution description file
write               writes a distribution description file
load                loads distribution weights from a file
save                saves distribution weights into a file
createAccus         creates accumulators for all distributions
clearAccus          clear accumulators for all distributions
loadAccus           loads distribution accumulators from a file
saveAccus           saves distribution accumulators into a file
map                 map all distributions
dist                measure distance between distributions
cat                 cat two distrib and their codebooks behind each other
cdcn                 CDCN: Codeword Dependent Cepstral Normalization (Logs-
pec not Cepstral Version!)
makeBBI             create BBI trees
freeBBI             destory BBI trees
loadBBI             load BBI trees from a file
saveBBI             save BBI trees from a file

**********************************************************************
DistribStream

DESCRIPTION
Distribution based stream


METHODS
puts
index               returns indices of named distributions
name                returns names of indexed distributions
get                 returns a distribution given a tagged phone sequence
score               compute distribution score
accu                accumulate sufficient statistic
update              update distributions/codebook

**********************************************************************
Dscfg
```

```
DESCRIPTION
configuration of a distribution

METHODS
puts

***********************************************************************
Duration

DESCRIPTION
explicite duration model

METHODS
puts

***********************************************************************
DurationSet

DESCRIPTION
A 'DurationSet' object is an array of explicite duration models.

METHODS
puts               displays the contents of a duration set
add                add new duration model(s) to a duration set
delete             delete duration model(s) from a duration set
read               read a duration set from a file
write              write a duration set into a file
index              return index of named duration model(s)
name               return the name of indexed duration model(s)
prob                 return the duration probability for a named duration
model
accu               accumulate training data
update             update the duration probabilities
dist               measure distance between duration models
scale              multiply all log-probs with given value
createAccus        allocate training data accumulators
freeAccus          allocate training data accumulators
loadAccus          load training data accumulators from file
saveAccus          save training data accumulators to file
clearAccus         clear training data accumulators
putsAccu           display training data accumulator

***********************************************************************
FArray

DESCRIPTION
Array of floats


METHODS
puts

***********************************************************************
FBMatrix
```

```
DESCRIPTION
Band matrix of float values


METHODS
puts               print matrix contents as TCL list
mel                melscale filterbank
meltri             triangular shaped melscale filterbank
meltra             trapezoid shaped melscale filterbank
display            display fbmatrix

***********************************************************************
FCovMatrix

DESCRIPTION
Covariance matrix type (float)

METHODS
puts
:=                 copy covariance matrix
+=                 add two scaled covariance matrices
variances          returns a list of the variances along the axis

***********************************************************************
FMatrix

DESCRIPTION
Matrix of float values


METHODS
puts               print matrix contents as TCL list
resize             resize matrix
copy               copy matrix
:=                 assign matrix (equiv. to 'copy')
get                get a single entry from a matrix
set                set a single entry in a matrix
DMatrix            convert from a DMatrix
clear              set all matrix values to 0
load               load matrix from file
bload              load matrix from binary file
bsave              save matrix to binary file
bappend            append matrix to binary file
trans              transpose matrix
add                a * matrixA + b * matrixB
dev                matrixA * matrixB
mulcoef            multiply each coefficient
mul                matrixA * matrixB
mulot              matrixA * matrixB'
bmulot             matrixA * bandmatrixB'
minmax             gives minimum and maximum
cosine             create cosine transformation matrix
neuralGas          neural gas clustering
cluster            create optimal codebook
display            display matrix
```

```
scatterPlot         scatter plot

*********************************************************************
FVector

DESCRIPTION
Vector of float values


METHODS
puts                print vector as TCL list
resize              resize vector
copy                copy vector
:=                  assign vector (equiv. to 'copy')
bload               load vector from binary file
bsave               save vector to binary file

*********************************************************************
Feature

DESCRIPTION
Feature


METHODS
puts

*********************************************************************
FeatureSet

DESCRIPTION
set of features


METHODS
puts
delete              delete a feature
name                get feature name for a given index
index               get feature index for a given name
display             displays a feature
SVector             insert SVector type object into feature set
FMatrix             insert FMatrix type object into feature set
setDesc             read a 'Feature Description'
setAccess           read a 'File Access Description'
readADC             read ADC file
writeADC            write ADC file
adc2pow             framebased power
peak                framebased peak distance
maxpeak             framebased maximum of peak to peak
zero                framebased zero crossing rate / sec
silTK               T.Kemp's silence feature
spectrum            framebased power spectrum
adc2mel             16 framebased melscale coefficients, 8 and 16 kHz only
corr                create puls in signals
puls                create puls in signals
tone                create audio signals
noise               create noise signal
xtalk               remove crosstalk with an adaptive filter
resample            resample audiosignal changing sampling rate
melscale            melscale from power spectrum
plp                 perceptual linear prediction
auditory            auditory filterbank
postaud             post processing for auditory filterbank
write               write feature file
read                read feature file
split               take coefficients <from> .. <to> of source feature
cut                 take frames <from> .. <to> of source feature
flip                take last frames first
delta               symmetrical delta coefficients: x(t+delta) - x(t-delta)
shift               shift frames: x(t+delta)
adjacent             put adjacent frames together: x(t-delta), x(t+1-delta),
..., x(t+delta)
lin                 m * source_feature + a
log                 m * log(source_feature + a)
alog                m * log(source_feature + b) with b=max/10^a
exp                 m * exp(a * source_feature)
pow                 m * (source_feature ^ a)
maxarg              index of maximum value per frame
meanarg             mean index per frame
aspike              remove spikes from signal
offset              remove offset adaptively from signal
filter              filter a feature
VTLN                Vocal Tract Length Normalization(VTLN)
normframe           normalize each frame
mean                calculate mean and variance
meansub             meansubtraction and variance normalisation
normalize           normalize coefficients to range <min> .. <max>
thresh               set coefficients to a specified value if they exceed a
threshold
beepSeg             segment (spectral) feature at beeper positions
silSeg              segment (spectral) feature at silence positions
add                 add two features: a * featureA + b * featureB
mul                 multiply two features: a * featureA * featureB
matmul              multiply matrix A with each frame x of feature: A * x
filterbank           multiply band matrix A with each frame x of feature: A
* x
concat              concat frames (or samples) of features
merge               merge coefficients (interleave samples) of features

TCL-DEFINED METHODS
eval                run feature description script (featureSetEval)
show                show feature set (featshow)
access               preprocess feature evaluation parameters (featureSetAc-
cess)
*********************************************************************
FlatFwd

DESCRIPTION
Search: Flat Forward Module
```

```
METHODS
puts

***********************************************************************
Forced

DESCRIPTION
Search: EXPERIMENTAL Beam Optimization Pass

METHODS
puts

***********************************************************************
HMM

DESCRIPTION
An 'HMM' object contains states, transitions and acoustic references

METHODS
puts                displays the contents of an HMM
make                create full detail HMM
lattice             create full detail HMM from a lattice

TCL-DEFINED METHODS
makeUtterance       create utterance HMM (hmmMakeUtterance)
***********************************************************************
Hypo

DESCRIPTION
Hypo is a subtype of HypoList only.
One Hypo contains a single hypothesis.
What you need is always a list of hypotheses.
Use HypoList to create such a list


METHODS
puts                -style normal|verbose|tcl -id <hypoid>

***********************************************************************
HypoList

DESCRIPTION
The object HypoList contains a list of hypotheses.


METHODS
puts                -style normal|verbose|tcl -id <uttid>
setPublish          same as puts for the moment
append              append hypo to hypolist

TCL-DEFINED METHODS
publish              produce output -- format specified by setpublish  (hy-
poListPublish)
***********************************************************************
IArray
```

```

DESCRIPTION
Array of integers


METHODS
puts

***********************************************************************
IMatrix

DESCRIPTION
Matrix of integer values


METHODS
puts                print matrix contents as TCL list
resize              resize matrix
clear               set all matrix values to 0
copy                copy matrix
:=                  assign matrix (equiv. to 'copy')
get                 get a single entry from a matrix
set                 set a single entry in a matrix
bload               load matrix from binary file
bsave               save matrix to binary file

***********************************************************************
LDA

DESCRIPTION
LDA


METHODS
puts
index               returns indices of named LDA classes
name                returns names of indexed LDA classes
add                 add a new LDA class to the set
delete              remove LDA class from the set
map                 add/get index to class mapping information
accu                accumulate samples from a path object
update              update the scatter matrices
clear               clear means
saveMeans           save means to a file
saveScatter         save scatter matrix to a file
loadMeans           load means from a file
loadScatter         load scatter matrix from a file

***********************************************************************
LDAClass

DESCRIPTION
LDA class

METHODS
```

```
puts

**********************************************************************
LatNode

DESCRIPTION
Lattice Node

METHODS

**********************************************************************
Lattice

DESCRIPTION
Lattice

METHODS
puts
add                add lattice nodes/links
hmm                Lattice to HMM conversion
rescore            Lattice trigram rescoring
rescoreNG          Lattice N-gram  rescoring
findNBest          create N-best list
write              write lattice to file
prune              prune lattice

**********************************************************************
List

DESCRIPTION
List of indexed items


METHODS
puts
name               translate indices to names
index              translate names to indices
delete             remove distribution from the set

**********************************************************************
Lm

DESCRIPTION
The object Lm contains a language model.


METHODS
puts               write out info about lm in tcl-format
score              get language model scores
debug              show internal data structures
activate           make this lm the current one
cachelines         change number of cachelines
utterance_context  change current utterance_context

**********************************************************************
```

```
MLAdapt

DESCRIPTION
Maximum Likelihood Adaptation


METHODS
puts
add                add items to the adaptation
clear              remove all items from the adaptation list
cluster            cluster items in the list
update             update codebook means
variance           update codebook variances

**********************************************************************
MLAdaptItem

DESCRIPTION
MLAdaptItem


METHODS
puts

**********************************************************************
MLNorm

DESCRIPTION
MLNorm


METHODS
puts
index              returns indices of named MLNorm classes
name               returns names of indexed MLNorm classes
add                add a new MLNorm class to the set
delete             remove MLNorm class from the set
map                add/get index to class mapping information
accu               accumulate samples from a path object
clear              clear accumulator
transform          transform samples along a path object

**********************************************************************
MLNormClass

DESCRIPTION
MLNorm class

METHODS
puts

**********************************************************************
ModelArray

DESCRIPTION
```

```
Array of models.

METHODS
puts                print model array
clear               remove all entries from the array
add                 add another model to the array

**********************************************************************
PTree

DESCRIPTION
Polyphonic Tree


METHODS
puts
add                 adds another polyphone to the tree
get                 find polyphone in the tree
models              returns a model array of models in the tree
split               split a tree by asking a question
question            find a question for splitting

**********************************************************************
PTreeNode

DESCRIPTION
PTreeNode


METHODS
puts

**********************************************************************
PTreeSet

DESCRIPTION
A 'PTreeSet' object is a set of polyphone context trees.

METHODS
puts                displays the contents of a PTreeSet object
add                 adds another polyphonic tree
name                find name of a polyphone tree
index               find index of a polyphone tree
read                reads polyphone tree from a file
write               writes polyphone tree to a file

**********************************************************************
Path

DESCRIPTION
A 'Path' object is filled by a forced alignment function and is used by
training functions

METHODS
puts                displays the contents of a path
labels              displays the contents of a path as labels
words               displays the word/variant labels
phones              displays the phones labels
reset               remove all items from a path
make                creates a path
viterbi             compute a Viterbi path for a given HMM
fwdBwd              compute a forward backward path for a HMM
lscore              compute the local scores
stateMatrix         matrix of state gamma scores
senoneMatrix        matrix of senone gamma scores
phoneMatrix         matrix of cum. phone gamma scores
wordMatrix          matrix of cum. word gamma scores
map                 map senone indices
bload               binary load of path items
bsave               binary save of path items

**********************************************************************
PathItem

DESCRIPTION
PathItem


METHODS
puts

**********************************************************************
PathItemList

DESCRIPTION
PathItemList


METHODS
puts
add                 add items to the path list
clear               remove all items from the path list

**********************************************************************
Phone

DESCRIPTION
Phone


METHODS
puts

**********************************************************************
PhoneGraph

DESCRIPTION
PhoneGraph
```

```
METHODS
puts
make               create PhoneGraph
build              create PhoneGraph from WordGraph

***********************************************************************
Phones

DESCRIPTION
A 'Phones' object is an array of strings, each of which is a phoneme.

METHODS
puts               displays the contents of a phone-set
add                add new phone(s) to a phone-set
delete             delete phone(s) from a phone-set
read               read a phone-set from a file
write              write a phone-set into a file
index              return index of named phone(s)
name               return the name of indexed phone(s)

***********************************************************************
PhonesSet

DESCRIPTION
A 'PhonesSet' object is a set of 'Phones' objects.

METHODS
puts               displays the contents of a set of phone-sets
add                add new phone-set to a set of phones-set
delete             delete phone-set(s) from a set of phone-sets
read               read a set of phone-sets from a file
write              write a set of phone-sets into a file
index              return index of named phone-set(s)
name               return the name of indexed phone-set(s)

***********************************************************************
Question

DESCRIPTION
A 'Question' object is a definition of a single question.

METHODS
puts               display one single question
answer             answer a question for a named context

***********************************************************************
QuestionSet

DESCRIPTION
A 'QuestionSet' object is a set of characteristic function definitions and
a set of questionSet.

METHODS
puts               displays the contents of a questionSet object
add                add a new question to a questionSet object
```

```
delete             remove a question from a questionSet object
read               read questionSet from a file
write              write questionSet into a file
index              return the index of a named question
name               return the name of an indexed question

***********************************************************************
Rewrite

DESCRIPTION
Rewrite Rule


METHODS
puts

***********************************************************************
RewriteSet

DESCRIPTION
Set of rewrite rules


METHODS
puts
add                add a new rewrite rule to the set
delete             remove rewrite rule from the set
read               reads a rewrite rules file
write              writes a rewrite rules file

***********************************************************************
SVector

DESCRIPTION
Vector of short values


METHODS
puts               print vector as TCL list
set                set single coefficient
resize             resize vector
copy               copy vector
:=                 assign vector (equiv. to 'copy')
lin                a * vector + b
add                a * vectorA + b * vectorB
minmax             gives minimum and maximum
power              gives the power value
mean               gives the mean value
histogram          histogram
display            display vector

***********************************************************************
SampleSet

DESCRIPTION
```

```
containers for samples

METHODS
puts
index              returns indices of named SampleSet classes
name               returns names of indexed SampleSet classes
add                add a new SampleSet class to the set
delete             remove SampleSet class from the set
map                add/get index to class mapping information
showmap            display class mapping information
accu               accumulate samples from a path object
clear              clear accumulation buffers
flush              flush accumulation buffers to file

***********************************************************************
SampleSetClass

DESCRIPTION
a class in a SampleSet

METHODS
puts
clear              clear accumulation buffers
flush              flush accumulation buffer to file

***********************************************************************
Search

DESCRIPTION
Search Module

METHODS
puts
treeInfo           get lm-lookahead info for a word
treeFwd            run tree forward pass of search
flatFwd            run flat forward on likely words
fullFlatFwd        run flat forward with all words
forced             run forced recognition
lattice            build lattice from last search pass

***********************************************************************
Senone

DESCRIPTION
Senone


METHODS
puts
stream             set stream weights

***********************************************************************
SenoneSet

DESCRIPTION
```

```
Set of senones


METHODS
puts
name               returns names of indexed senones
add                add a new senone to the set
get                find a senone given phonetic context
read               reads a senone description file
write              writes a senone description file
score              compute the score for a senone and a frame
scores             compute the scores for a list of senones
entropy            compute frame entropy and a priori probability
accu               accumulate training data for the given path
update             update the underlying acoustic parameters
stream             set stream weights
reset              reset senoneSet
engage             engage in warp speed (make process fork)
every              make every child evaluate a Tcl command

***********************************************************************
SenoneTag

DESCRIPTION
SenoneTag


METHODS
puts

***********************************************************************
StateGraph

DESCRIPTION
StateGraph


METHODS
puts
build              create StateGraph from PhoneGraph

***********************************************************************
Tag

DESCRIPTION
Tag


METHODS
puts               print information about tag

***********************************************************************
Tags

DESCRIPTION
```

```
A 'Tags' object is an array of strings.

METHODS
puts               displays the contents of a tags-set
add                add new tag(s) to a tags-set
delete             delete tag(s) from a tags-set
read               read a tag-set from a file
write              write a tag-set into a file
index              return index of named tag(s)
name               return the name of indexed tag(s)

***********************************************************************
TmSet

DESCRIPTION
A TmSet is a set of state transition model objects (Tm)


METHODS
puts               displays the contents of a transition model
add                add a Tm to the list
read               reads a TmSet from a file
write              writes a TmSet to a file
index              return index of named Tm(s)
name               return the name of indexed Tm(s)

***********************************************************************
Topo

DESCRIPTION
A 'Topo' object is a definition of a single topology description.

METHODS
puts               display one single topo

***********************************************************************
TopoSet

DESCRIPTION
A 'TopoSet' object is a set of different topologies.

METHODS
puts               displays the contents of a TopoSet object
add                add a new topo to a TopoSet object
delete             remove a topo from a TopoSet object
read               read TopoSet from a file
write              write TopoSet into a file
index              return the index of a named topo
name               return the name of an indexed topo

***********************************************************************
Tree

DESCRIPTION
A 'Tree' object is an allophone clustering tree.
```

```
METHODS
puts               displays the contents of a tree object
add                add a new node to the tree
read               read a tree from a file
write              write a tree into a file
list               list a tree contents in TCL list format
get                descend a tree for a given phone sequence
trace              trace a tree for a given phone sequence
index              return the index of a node
name               return the name of an indexed node
question           return best splitting question to ask
split              split node according to a question

***********************************************************************
TreeFwd

DESCRIPTION
Search: Tree Forward Module

METHODS
puts

***********************************************************************
TreeNode

DESCRIPTION
TreeNode


METHODS
puts

***********************************************************************
Vocab

DESCRIPTION
A Vocab is the list of words the recognizer can recognize
To build a Vocab instance you have to specify:
vocablist       filename for vocabulary
-fillerlist     filename for fillers
-dictionary     phoneme dictionary object to use for search
-acousticModel  acoustic model to use for search


METHODS
activate           activates indexed vocab entry
deactivate         deactivates indexed vocab entry
puts               displays the contents of a vocabulary
index              returns indices of named vocab words
name               returns names of indexed vocab entries

***********************************************************************
Word
```

```
DESCRIPTION
Word with tagged phone transcription


METHODS
puts

************************************************************************
WordGraph

DESCRIPTION
WordGraph


METHODS
puts
make                   create WordGraph
lattice                create WordGraph from lattice

************************************************************************
XWModel

DESCRIPTION
blah

METHODS
puts                   displays the contents of an xword model

************************************************************************
XWModelSet

DESCRIPTION
set of blah

METHODS
puts                   displays the contents of an xwmodel set
add                    add a state graph to a set
```

# Literaturverzeichnis

[1] *Alfred, Ying Pang; Chan, L.W.; Ching, P.C.*: Automatic Recognition of Continuous Cantonese Speech with Very Large Vocabulary, EuroSpeech '97, Rhodos (Volume 3 pages 1551 – 1554)

[2] *Baoyong, Sun; Feng, Chen; Jianying, Cui; Xiuli, Bing*: Learning Chinese Through Group-Character Analyses (分沮识字学汉语) Band 1 - 3, Sinolingua Peking 1993

[3] *Belew, Peter*: Programming Windows 95 Unleashed, SAMS Publishing 1995

[4] *Binyong, Yin*: Modern Chinese Characters (现代汉子), Sinolingua Peking 1994

[5] Brudermüller, Susanne; 许素芳，施其南 *et al.*: Handwörterbuch Deutsch-Chinesisch Chinesisch-Deutsch (精选德汉汉德词典), The Commercial Press, Peking und Langenscheidt KG, Berlin und München, 1994

[6] *Burda, Arthur; Bertelsons, Boris*: Soundblaster, Data Becker GmbH 1994

[7] *Chen, C.J.; Gopinath, R. A.; Monkowski, M.D.; Picheny, M.A.; Shen, K.*: New methods in Continuous Mandarin Speech Recognition, EuroSpeech '97, Rhodos (Volume 3 pages 1543 - 1546)

[8] Chinese, Star 2.0 (中文之星), SunTendy 1994, http://www.chinesestar.com/, http://www.suntendy.com.cn/

[9] *Dejin, Li; Meizhen, Cheng*: Praktische chinesische Grammatik für Ausländer (外国人实用汉语语法), Sinolingua Peking 1993

[10] *Finke, Michael; Geutner, Petra; Hild, Hermann; Kemp, Thomas; Ries, Klaus; Westphal, Martin*: The Karlsruhe-Verbmobil Speech Recognition Engine, ICASSP 97, Munich

[11] *G. Fant*: The Acoustic Theory of Speech Production, Mouton & Co., The Hague, 1960

[12] *Gao, Yuqing; Hon, Hsiao-Wuren; Lin, Zhiwei; Loudon, Gareth; Yaganantha, S.;Yuan, Baosheng*: Tangerine: A Large Vocabulary Mandarin Dictation System, ICASSP 95, Detroit (Volume 1, Page 77)

[13] *Gernet, Jacques*: Die chinesische Welt, Insel Verlag 1989

[14] *Guder-Manitius, Andreas*: Chinesisch-Deutsches Lernwörterbuch, Verlag Ute Schiller, Berlin 1991

[15] *Haberland, Nils*; *Stephan, Kanthak; u.a.*: Report Spracherkennung, c't Magazin für Computertechnik 5/98

[16] *Han Ying Xuci Cidian* (汉英虚词词典): A Chinese-English Dictionary of Function Words, Sinolingua Peking 1992

[17] *Ho, Tai-Hsuan; Yang, Kae-Cherng; Huang, Kuo-Hsun; Lee, Lin-Shan*: Improved search strategy for large vocabulary continuous mandarin speech recognition, ICASSP 98, Seattle (Volume 2, Page 825, Paper number 2381)

[18] *Hongyi, Tao*; Tangshou, Zhao: Handwörterbuch der Gegenwartssprache (现代汉德德汉词典), Peking 1994

[19] IPA (1989): The International Phonetic Alphabet; JIPA (19,2), S.81f

[20] *Jelinek, Frederick*: Statistical Methods for Speech Recognition; The MIT Press; Cambridge, Massachusetts; London, England

[21] *Kano, Nadine*: Developing International Software, Microsoft Press 1995

[22] *Keller, Eric*: Fundamentals of Speech Synthesis and Speech Recognition; John Wiley & Sons; Chichester, New York, Brisbane, Toronto, Singapore 1994

[23] *Kruglinski, David J.*: Inside Visual C++, Microsoft Press Deutschland 1994

[24] *Lavie et al.*: JANUS-III: Speech-to-Speech Translation in Multiple Languages. ICASSP 1997, Munich.

[25] *Lee, Yue-Shi; Chen, Hsin-Hsi*: Using acoustic and prosodic cues to correct chinese speech repairs, EuroSpeech '97, Rhodos (Volume 4 pages 2211 - 2214)

[26] *Li, Aijun*: Perceptual study of intersyllabic formant transitions in synthesized V1-V2 in standard chinese, EuroSpeech '97, Rhodos (Volume 4 pages 2143 - 2146)

[27] *Lin, Sung-Chien; Chien, Lee-Feng; Chen, Ming-Chiuan; Lee, Lin-Shan; Chen, Ker-Jiann*: Intelligent retrieval of very large chinese dictionaries with speech queries, EuroSpeech '97, Rhodos (Volume 3 pages 1463 - 1466)

[28] *Lin, Sung-Chien; Tsai, Chi-Lung; Chien, Lee-Feng; Chen, Ker-Jiann; Lee, Lin-Shan:* Chinese language model adaption based on document classification and multiple domain-specific language models, EuroSpeech '97, Rhodos (Volume 3 pages 1463 - 1466)

[29] *Lyu, Ren-Yuan; u.a.*: Golden Mandarin (III) – A User-Adaptive Prosodic-Segment-Based Mandarin Dictation Machine for Chinese Language with Very Large Vocabulary, ICASSP 95, Detroit (Volume 1, Page 57)

[30] *Ma, Bin; Huang, Taiyi; Xu, Bo; Zhang, Xijun; Qu, Fei*: Context-Dependent Acoustic Models for Chinese Speech Recognition, ICASSP 96, Atlanta (Volume 1, Page 455)

[31] *Mayfield-Tomokiyo, Laura; Ries, Klaus*: An automatic method for learning a japanese lexicon for recognition of spontaneous speech: ICASSP 98, Seattle (Volume 1, Page 305, Paper number 2305)

[32] *Morgan, Nelson; Bourland, Herve*: An Introduction to Hybrid HMM/Connectionist Continuous Speech Recognition, International Computer Science Institute Berkeley, 1994

[33] *Neukirchen, Christoph; Rigoll, Gerhard*: Ein systematischer Vergleich von diskreten, kontinuierlichen und hybriden HMM-basierten Systemen zur Spracherkennung, Gerhard-Merctor-Universität- GH Duisburg, ~1995

[34] *Ney, Hermann*: The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing, April 1984

[35] *Ni, Jin-Fu; Wang, Ren-Hua; Hirose, Keikichi*: Quantitative Analysis and Formulation of Tone Concatenation in Chinese F0 Contours, EuroSpeech '97, Rhodos (Volume 1 pages 195 - 198)

[36] NJWIN, NJStar Software Corp & Hongbo Ni 1995-1997, http://www.njstar.com

[37] *Oertel, Helmut*: Hidden Markov Modelle – Theorie und Anwendungen, TH Berlin, Fachbereich Informatik, 1996 (http://www.cs.tu-berlin.de/~oertelh/markov.html)

[38] *Ousterhout, John K.*: Tcl and The Tk Toolkit, Addison –Wesley, 1994

[39] *Peihui, He; Wenhua, Xiong; Xiuxian, Mei*:Practical Chinese Reader (实用汉语课本) Book I - VI, The Commercial Press, Peking 1991

[40] *Peiyuan, Li; Yuan, Ren; Shuhua, Zhao; Prof. Zhao, Käthe*: Grundkurs der chinesischen Sprache (基础汉语课本) Band 1 - 4, Sinolingua Peking 1979-88

[41] *Petzhold, Charles*: Windows 95 Programmierung – Das definitive Entwicklerhandbuch zur Windows-95-API, Microsoft Press Deutschland 1996

[42] *Ping-Chien, Ly; Motsch, Monika*: Kurze Grammatik der modernen chinesischen Hochsprache, Dürr & Kessler Rheinbreitbach

[43] *Rabiner, Lawrence R.; Schafer, Ronald W.*: Digital processing of speech signals, Prentice-Hall, Inc.1978

[44] *Reichert, Jürgen:* Studienarbeit: Lautschriftumsetzung und Wortrennung der chinesischen Schriftsprache, Universität Karlsruhe1997

[45] RichWin, http://www.richwin.com/

[46] *Ries, Klaus:* A class based approach to domain adaption and constraint integration for empirical m-gram models, Eurospeech 97, Rhodos (Volume 4 pages 1983 - 1986)

[47] *Robinson, Tony*: Simple lossless and near-lossless waveform compression, Cambridge University Engineering Department 1994
http://www.softsound.com/DrTonyRobinson.html

[48] *Ruske, Günther*: Automatische Spracherkennung – Methoden der Klassifikation und Merkmalsextraktion, R. Oldenbourg Verlag München Wien 1994

[49] *Sagerer, Gerhard*: Automatisches Verstehen gesprochener Sprache, BI Wissenschafts-verlag 1990

[50] *Schukat-Talamazzini, E.G.:* Automatische Spracherkennung, Verlag Vieweg Braunschweig/Wiesbaden 1995

[51] *Schultz, Tanja; et al.*: Language Independent Language Adaptive Large Vocabulary Speech Recognition, ICSLP 98, Sydney (to appear)

[52] *Schultz, Tanja; Waibel Alex*: Fast Bootstrapping of LVCSR Systems with Multilingual Phoneme Sets

[53] *Schultz, Tanja; Westphal, Michael; Waibel, Alex*: The GlobalPhone Project: Multilingual LVCSR with Janus3 2^nd SQEL-Workshop, Plzen, Tschechien 1997.

[54] *Shen, Jia-lin; Lee, Lin-shan*: Fast and Accurate Recognition of Very-Large-Vocabulary Continous Mandarin Speech for Chinese Language with Improved Segmental Probability Modeling, ICASSP 96, Atlanta (Volume 1, Page 125)

[55] *Stermann, Klaus; Reiser-von Loh, Katherin; Wan-yee, Ho; Jian-ming,Li; Yong-xin Zhao*: Langenscheidts Sprachführer Chinesisch, Langenscheidt KG, Berlin und München 1994

[56] *Stolz, Axel*: Das große Soundblaster Buch, Data Becker GmbH 1994

[57] The IPA 1989 Kiel Convention. In: Journal of the Internatioal Phonetic Association 1989(19) S. 67-82

[58] *Toth, Viktor*: Visual C++ 5 – Das Kompendium, Markt & Technik Buch- und Software-Verlag GmbH 1997

[59] TwinBridge, http://www.twinbridge.com/

[60] Union Way, http://www.unionway.com/

[61] *Via Voice 4.3 Gold:* http://www.software.ibm.com/is/voicetype/

[62] *Waibel, Alex; Hampshire, John*: Building Blocks for Speech – Modular neural networks are a new approach to high-performance speech recognition, BYTE 8/1989

[63] *Waibel, Alex; Lee, Kai-Fu:* Readings in Speech Recognition, Morgan Kaufmann Publishers, Inc. San Mateo, California

[64] *Wang, Chao; Glass, James; Meng, Helen; Polifroni, Joe; Seneff, Stefanie; Zue, Victor:* Yinhe: A mandarin chinese version of the galaxy system, EuroSpeech ’97, Rhodos (Volume 1 pages 351 - 354)

[65] *Wang, Hsin-min; Shen, Jia-lin; Yang, Yen-ju; Tseng, Chiu-yu; Lee, Lin-shan*: Complete recognition of continous mandarin speech for chinese language with very large vocabulary but limited training data, ICASSP 95, Detroit (Volume 1, Page 61)

[66] *Wang, Yih-Ru; Chen, Sin-Horng*: Mandarin telephone speech recognition for automatic telephone number directory service, ICASSP 98, Seattle (Volume 2, Page 841, Paper number 1353)

[67] *Westphal, Martin*; *Schultz, Tanja; Waibel, Alex*: Linear Discriminant – A New Criterion for Speaker Normalization, to appear ICSLP 1998, Sydney

[68] *Xinhua Zidian* (新华词典); Peking 1975

[69] *Yang, Kae-Cherng; Ho, Tai-Hsuan; Chien Lee-Feng; Lee, Lin-Shan*: Statistics-based segment pattern lexicon – a new direction for chinese language modeling, ICASSP 98, Seattle (Volume 1, Page 169, Paper number 2395)

[70] *Yezhi, Yang; Lilin, Zhai*; *Pan, Zaiping et al.*: Deutsch-Chinesisches Wörterbuch (德汉词典), Shanghai 1987.

[71] *Young, Steve*: Large Vocabulary Continuous Speech Recognition – a Review, Cambridge University Engineering Department, Cambridge, 1996

[72] *Zell, Andreas*: Simulation Neuronaler Netze, Stuttgart 1996

[73] *Zeppenfeld, Torsten; Finke, Michael; Ries, Klaus; Westphal, Martin; Waibel, Alex*: Recognition of Conversational Telephone Speech using the Janus Speech Engine, ICASSP 97, Munich

[74] *Zhan, Puming; Wegmann, Steven; Lowe, Steve*: Dragon Systems‘ 1997 Mandarin Broadcast news System, Darpa (Broadcast News Transcription and Understanding Workshop) 98, Lansdowne, Virginia

[75] *Zhang, Jin-song, Hirose Keikichi:* A robust tone recognition method of chinese based on sub-syllabic F0 contours, ICSLP 98, Sydney

[76] *Zhang, Shuwu; Huang, Taiyi*: An Integrated Language Modeling with n-gram model and WA model for Speech Recognition, EuroSpeech ’97, Rhodos (Volume 5 pages 2699 - 2702)

[77] *Zhenmin, Xu; Huiying, Chen; Dr. Rainer Kloubert et al.*: Das neue Chinesisch-Deutsche Wörterbuch (新汉德词典), Peking 1993.

[78] *Zhou, Liang; Imai, Satoschi*: Chinese all syllables recognition using combination of multible classifiers, ICASSP 96, Atlanta (Volume 6, Page 3495)

[79] *Zu, Yiqing:* Sentence design for speech synthesis and speech recognition database by phonetic rules, EuroSpeech ’97, Rhodos (Volume 2 pages 743 - 746)